COMP90082-2024-FL-Koala Home

Project Overview

We are students from the University of Melbourne, undertaking a project for our client, Austin Health. The Radiation Oncology Department at Austin Health is in need of a Python script execution website to streamline the execution, management, and version control of Python scripts used for tasks such as analyzing CT and MRI images and managing patient data.

Our goal is to create a centralized, user-friendly platform that enhances operational efficiency and accessibility for all team members, including those without Python knowledge. By implementing this solution, we aim to eliminate the need for local Python installations, simplify IT management, and ensure secure, efficient script execution. This platform will provide universal access, robust version control, and a secure environment, ultimately improving the department's ability to deliver high-quality care using advanced technology.

Supervisor: Wei Wang

Team member:

ID	Name	Role
1367102	Zhihao Liang	Quality Assurance
1351342	Lin Duan	Project Owner
1397061	Lingyi Kong	Scrum Master
1373110	Yuncong Ji	Archtecture Leader
1133093	Junhao Kong	Deployment Leader
1132416	Yijun Liu	Development Leader

Table of Content

Quick access to other resources

Github: https://github.com/COMP90082-2024-SM1/FL-Koala Connect your Github account

Trello: Trello Agile Sprint Board - FL

Client shared document: FL-Project - Google Drive

Project Overview

The project aims to streamline the management and execution of Python scripts used in radiation oncology at ONJ, a department within Austin Health. These scripts facilitate various tasks related to treating cancer patients using imaging devices and a linear accelerator for radiation delivery. The current process involves manual execution of scripts, posing challenges in version control, accessibility, and maintenance.

To address these issues, the project proposes two main components:

1. **Prototype Web Interface for Script Execution:** Utilizing Flask or a suitable alternative, a web-based dashboard will be developed. This interface will allow users to execute scripts without the need for Python installation. It aims to simplify the execution process, particularly for non-Python users, and enable centralized access to scripts.
2. **GitHub Repository for Version Control:** The project will organize the scripts into a GitHub repository, enabling version control, collaboration, and centralized access. GitHub's features, such as branching, pull requests, and user management, will facilitate efficient script management and development.

In this section, we provide a detailed background description, outline the motivation for this project, and discuss the personas of potential users.

Background Description

Motivation and Goals

Personas

Quick access to other resources

Github: https://github.com/COMP90082-2024-SM1/FL-Koala Connect your Github account

Trello: Trello Agile Sprint Board - FL

Client shared document: FL-Project - Google Drive

Background Description

Radiation oncology department within Austin Health represents a critical intersection of medical expertise and technological innovation in cancer treatment. With the ever-evolving landscape of oncological research and technology, the department continually seeks to optimize its processes to deliver the best possible care to patients.

Central to the treatment process are advanced imaging techniques such as CT and MRI, which provide crucial insights into the location and extent of tumors within patients' bodies. These images serve as the foundation for treatment planning, where precise calculations and simulations are performed to determine the optimal delivery of radiation to target cancerous tissues while minimizing damage to healthy surrounding tissue.

The linchpin in this treatment paradigm is the linear accelerator, a sophisticated piece of machinery that generates high-energy radiation beams used to eradicate cancer cells. The precise calibration and delivery of these beams are guided by complex algorithms and calculations, which are often facilitated by custom Python scripts.

Given the inherently interdisciplinary nature of radiation oncology, the department comprises individuals with diverse skill sets and backgrounds, ranging from medical professionals to technologists and researchers. This diversity presents both challenges and opportunities in terms of streamlining processes and ensuring seamless collaboration.

Historically, the execution of Python scripts within the department has been a decentralized and ad-hoc affair. Scripts are often stored on individual workstations or shared drives, making version control and access management cumbersome tasks. Furthermore, the reliance on Python as the scripting language presents challenges for non-technical users who may lack the necessary environment setup to execute these scripts.

The need for a centralized, user-friendly solution to script management and execution becomes evident in this context. By consolidating scripts into a GitHub repository and providing a web-based interface for their execution, the department aims to overcome these challenges and unlock new efficiencies in its workflow.

Moreover, the decision to leverage GitHub as the version control platform aligns with broader trends in collaborative software development and open science initiatives. GitHub's robust features for code management, issue tracking, and collaboration make it an ideal choice for fostering transparency and reproducibility within the department's research and development efforts.

In summary, the background of the project underscores the critical role of Python scripts in the radiation oncology workflow and the imperative to modernize and streamline their management and execution. By embracing web technologies and version control best practices, the department seeks to empower its team members to work more efficiently and collaboratively towards the common goal of advancing cancer care.

Motivation and Goals

Motivation

The primary motivation behind developing this Python script execution website for the Radiation Oncology Department at Austin Health is to streamline their workflow and enhance their operational efficiency. The department treats cancer patients using advanced imaging devices, such as CT and MRI, combined with a linear accelerator to deliver high-energy radiation. Given the complexity of treating conditions hidden inside the body with invisible methods, the department frequently relies on small Python scripts to perform minor yet essential tasks, including copying files and editing text files.

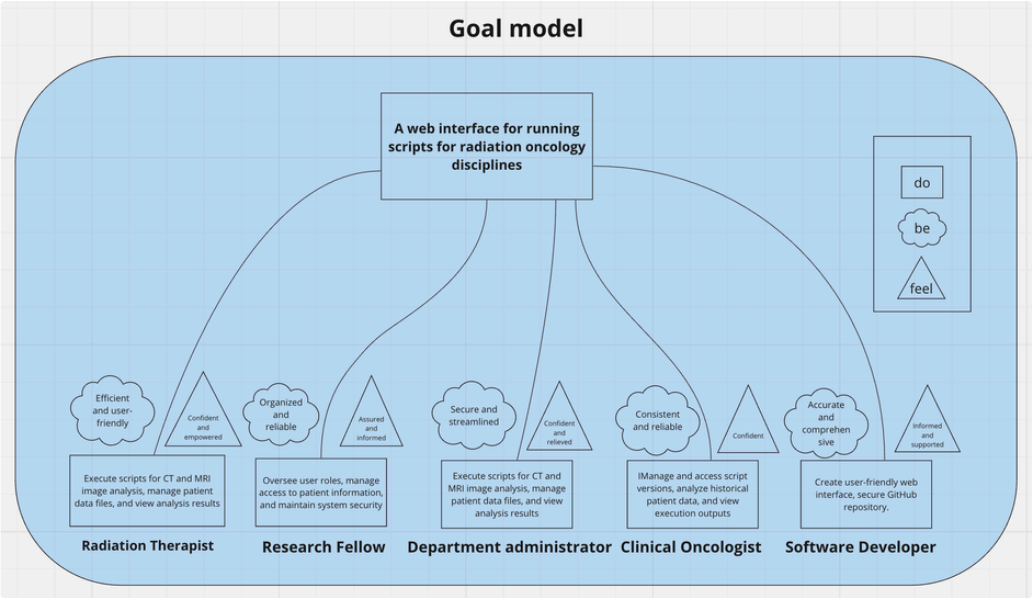
Currently, various team members can edit and access these scripts, but the lack of a centralized system creates challenges in monitoring versions and ensuring consistent script execution, especially for non-Python users who do not have Python installed and find remote access cumbersome. Our goal is to provide a robust, user-friendly solution that allows team members to run scripts from any device and location without needing prior Python knowledge or installation. Additionally, the solution will simplify IT management and script organization, improving overall efficiency and collaboration.

do / be / feel

Roles	Do(Functional Goal)	Be(Quality Goal)	Feel(Emotional Goal)
Radiation Therapist (Dr. Laura Kim)	Execute scripts for CT and MRI image analysis, manage patient data files, and view analysis results	Efficient and user-friendly	Confident and empowered
Research Fellow (Dr. Adam Jackson)	Manage and access script versions, analyze historical patient data, and view execution outputs	Organized and reliable	Assured and informed
Department administrator (Dr. Andy Tina)	Oversee user roles, manage access to patient information, and maintain system security	Secure and streamlined	Confident and relieved
Clinical Oncologist (Dr. Patric Wang)	Review analyzed patient data and make clinical decisions based on script outputs	Consistent and reliable	Confident
Software Developer (John Alex)	Integrate with medical databases and ensure compatibility	Accurate and comprehensive	Informed and supported

Goals

1. **Universal Access:** Enable team members to execute Python scripts effortlessly from any device and location, eliminating the need for local Python installations.
2. **Centralized Management:** Provide a centralized platform for running, managing, and version-controlling scripts, ensuring consistency and ease of access.
3. **IT Efficiency:** Simplify IT department tasks related to script management and user permissions, enhancing overall system administration.
4. **User-Friendly Interface:** Develop an intuitive web interface that allows non-Python users to run scripts without technical barriers.
5. **Version Control:** Implement a robust version control system using GitHub or a suitable alternative to track and manage script changes effectively.
6. **Security:** Ensure the platform adheres to strict security standards to protect sensitive patient data and script integrity.



Scope

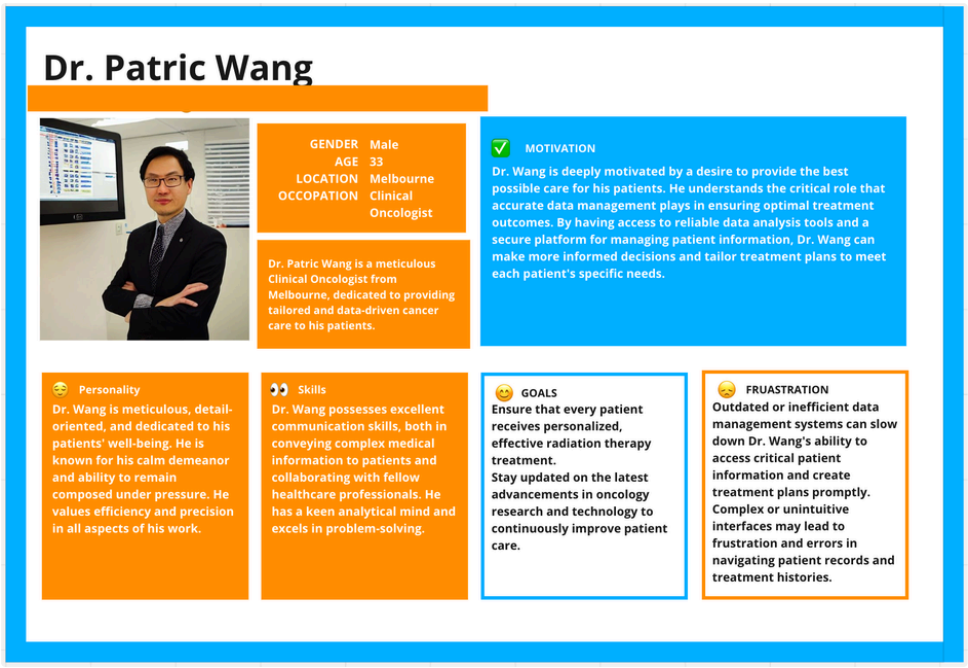
Based on the product backlog, the scope of the project includes the following:

1. **Script Execution:**
   - Develop a web interface to run Python scripts for various tasks, such as CT and MRI image analysis.
   - Ensure the interface is user-friendly and accessible to non-Python users.
   - Support the execution of scripts from any device and location.
2. **File Management:**
   - Implement functionality to support multiple file or folder uploads for analysis.
   - Provide a mechanism to obtain and display script output efficiently.
   - Allow users to save and retrieve frequently used scripts.
3. **Script Management:**
   - Create a dashboard for easy script execution and management.
   - Provide detailed usage instructions for each script to assist non-technical users.
   - Enable version control for all scripts, utilizing GitHub or an alternative.
4. **User and Role Management:**
   - Implement role-based access control to differentiate between administrators and normal users.
   - Ensure administrators can manage user accounts, including password resets and user information updates.
5. **Integration and Compatibility:**
   - Integrate the website with SQL-based databases for consistent data management.
   - Ensure smooth integration with the hospital's existing clinical data systems.
6. **Execution Monitoring:**
   - Develop a dashboard to display the status of script executions.
   - Implement notifications to inform users when long-running scripts have completed.
   - Provide access to execution history for reference and review.
7. **Prototype Development:**
   - Build a prototype using Flask or an alternative framework, considering feedback and potential improvements.
   - Plan for hosting the prototype on the hospital's system with clinical data for real-world testing and validation.

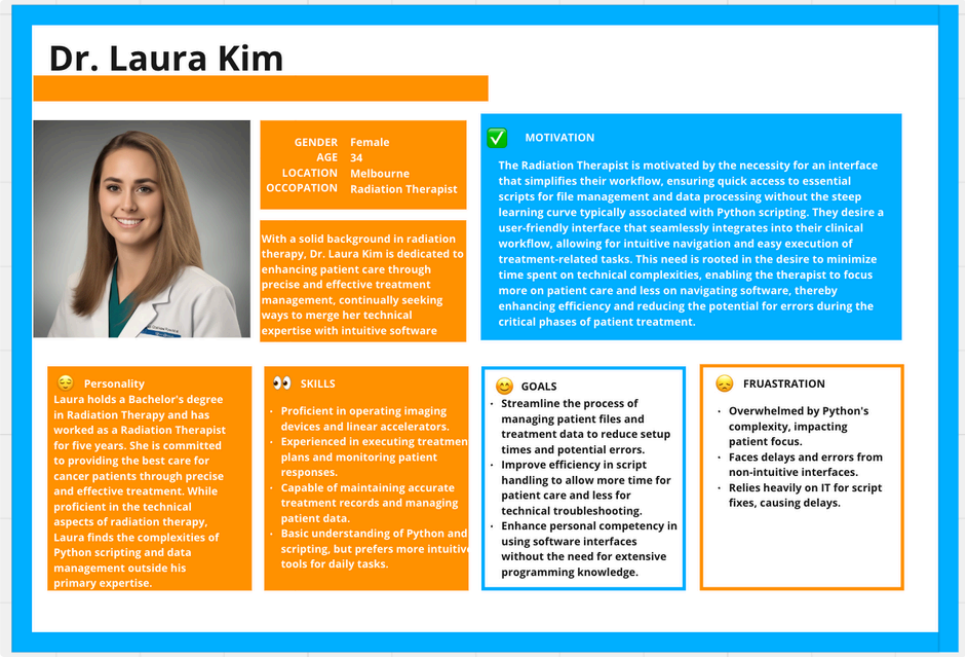
By addressing these areas, we aim to deliver a comprehensive solution that enhances the operational capabilities of the Radiation Oncology Department, making their workflow more efficient, accessible, and manageable.

Personas

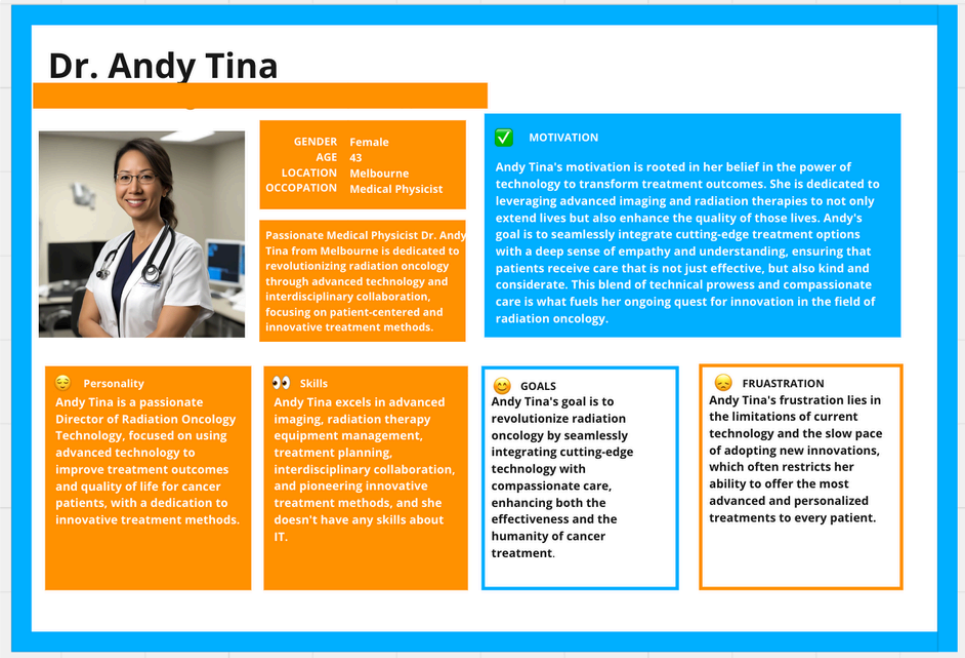
1. **Clinical Oncologist (Dr. Patric Wang)**
   - Background: Experienced oncologist specializing in radiation therapy.
   - Role: Oversees patient treatment plans and relies on accurate data management.
   - Needs: Access to scripts for data analysis and management, easy-to-use interface, assurance of data security and integrity.
   -



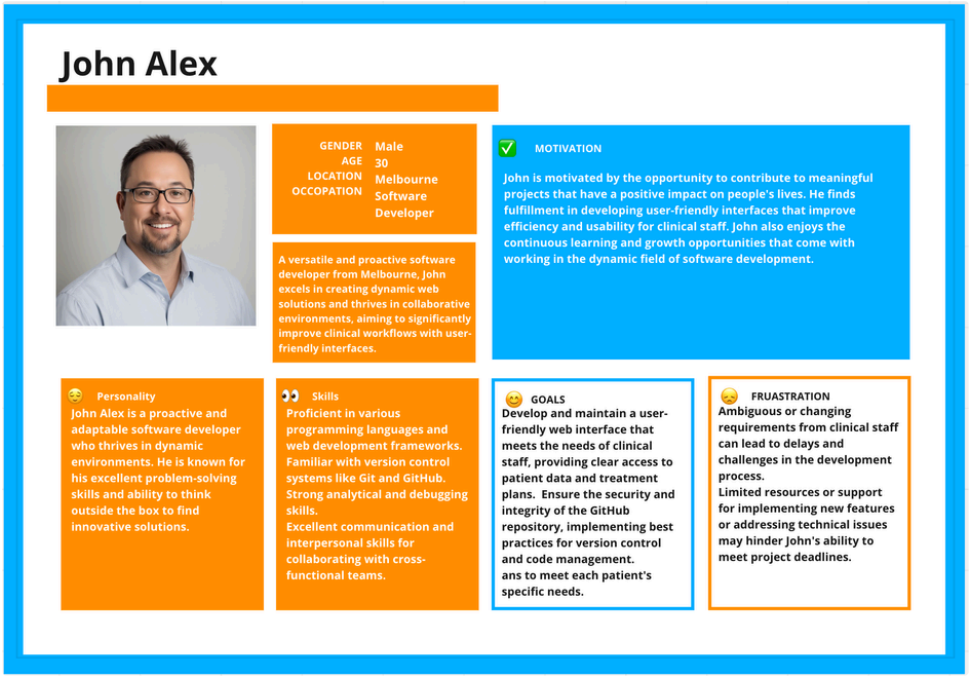
2. **Radiation Therapist (Dr. Laura Kim)**
   - Background: Clinical staff responsible for operating imaging devices and linear accelerators.
   - Role: Executes treatment plans and manages patient data during therapy sessions.
   - Needs: Quick access to scripts for file management and data processing, user-friendly interface compatible with clinical workflow.
   -



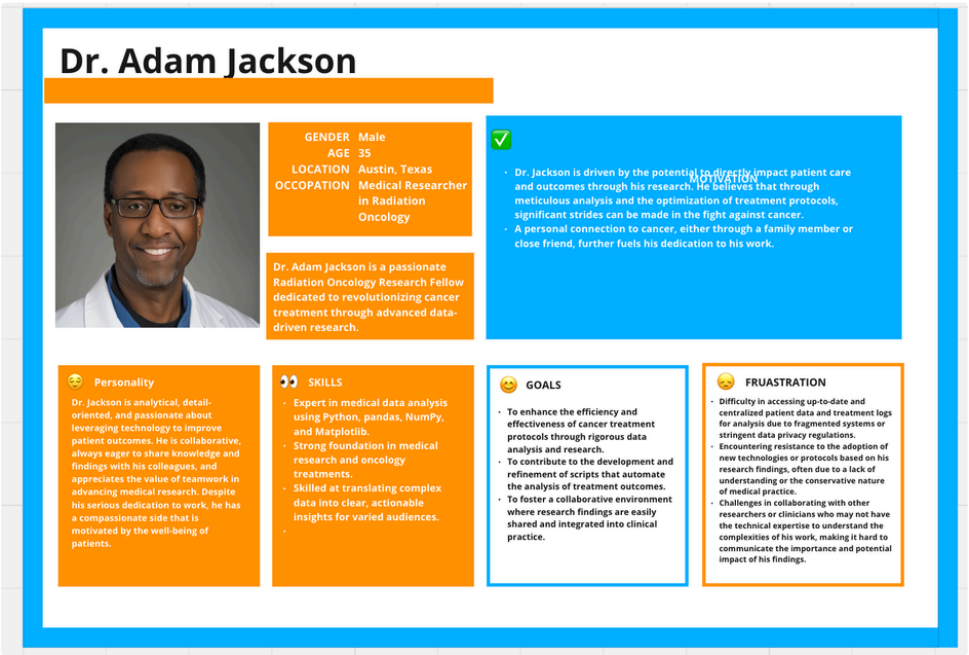
3. **Medical Physicist (Dr. Andy Tina)**
   - Background: Physics expert specializing in radiation therapy technology.
   - Role: Ensures the safe and accurate delivery of radiation doses to patients.
   - Needs: Version control for scripts, integration with existing hospital systems, ability to customize scripts for specific research or clinical needs.
   -



4. **Software Developer (John Alex)**
   - Background: IT professional with experience in software development.
   - Role: Develops and maintains the web interface and GitHub repository.
   - Needs: Clear requirements from clinical staff, flexibility to adapt to changing needs, collaboration tools for team communication and project management.
   -



5. **Research Fellow (Dr. Adam Jackson)**
   - Background: Medical researcher interested in analyzing treatment outcomes and optimizing protocols.
   - Role: Conducts research projects using patient data and treatment logs.
   - Needs: Access to scripts for data analysis and visualization, ability to contribute to script development and share findings with the team.



Requirements

In this section, email comminucation within client and team are loged. Different version of product backlog within each sprint is documented for further reference.

Client Email Log

Product backlog

How to run scripts shared by the client

Client Email Log

In this page, it includes all email communication within team and client. Each email constains further clarification about project requirements. And discussed topics are included in the following table for further reference.

Time	Discussed topic	Email Log
March 23	• Various scripts expected? • Suceessful running prompt? • Change the absolute path for those scripts.	 **FL-Project clien…ify.pdf** 23 May 2024, 10:49 AM
March 23	• System access control	 **FW_ Clarificati… 082.pdf** 23 May 2024, 10:51 AM
April 3	• Product - website or desk app?	 **FL-Product For… ion.pdf** 23 May 2024, 11:04 AM
April 19	• User story validation	 **FL-User Story V…ion.pdf** 23 May 2024, 11:05 AM
May 10	• Script path • Script iteration	 **FL-Koala-Script…ion.pdf** 23 May 2024, 11:06 AM
June 4	• Handover section	 **The University … ala.pdf** 07 Jun 2024, 05:34 AM

Product backlog

The product backlog (user story) may change during the process of developing, as we have more comminication with client and approach client for clarification for developing details. The feedback from client for our product demo, suggestion from supervisor may also influence the content and structure of product backlog.

It is recommended to adapt as last version of product backblog as reference.

User Story (Sprint 1 Version)

Owned by Chloe_Duan • Updated on May 23, 2024

Key content Summary User story classification Product backlog According to our clients needs, product backlog is implement with epics: script execution and monitoring; script management and integration, and user access and security, and user experience and dashboard design. Product backlog is designed including prioity and size estimation, which ca

Confluence Open preview

User Stories (Sprint 2 Version)

Owned by Chloe_Duan • Updated on May 23, 2024

Key content Summary User story classification Product backlog According to our clients needs, product backlog is implement with epics: script execution and monitoring; script management and integration, and user access and security, and user experience and dashboard design. Product backlog is designed including prioity and size estimation, which ca

Confluence Open preview

User Stories (Sprint 3 Version)

Owned by Chloe_Duan • Updated on May 23, 2024

Key content Summary User story classification Product backlog According to our clients needs, product backlog is implement with epics: script execution, script management, and user access and execution management. Product backlog is designed including prioity and size estimation, which can be considered as a requirement of project with acception cr

Confluence Open preview

User Story (Sprint 1 Version)

Key content

- Summary
- User story classification
- Product backlog

According to our clients needs, product backlog is implement with epics: script execution and monitoring; script management and integration, and user access and security, and user experience and dashboard design. Product backlog is designed including prioity and size estimation, which can be considered as a requirement of project with acception criteria for each user story. The classification of prioity and size estimation is defined as following.

User stories classification:

Size Estimation	
Small	User stories that can be completed in less or equal than 1 day by one person.
Medium	User stories that can be completed in 2 to 4 days by one person.
Large	User stories that can be completed in 5 to 7 days by one person.

Priority	
Must have	Non-negotiable features that must be implemented.
Should have	Important features that can add significant value.
Could have	Features that can add value but impact a little if left out.
Will not have	Features that are not a need for the project in this specific time-frame.

Product backlog

ID	Epic	User story	Priority	Size estimation	Acception Criteria
1	Script Execution and Monitoring	As a radiation therapist, I want the interface to automatically handle file paths when running scripts so that I can concentrate on patient care instead of software navigation.	Must have	Large	• Interface automatically detects and inputs file paths for scripts. • Therapist can execute scripts without manual path adjustments. • System verifies path accuracy before script execution.
2	Script Execution and Monitoring	As a medical physicist, I want to run a specific script from a web interface so that I can perform routine tasks	Must have	Large	• Web interface allows selection and execution of scripts. • No Python installation or command line knowledge needed.

		without needing to install Python or understand the command line.			• Interface provides immediate visual confirmation when script starts.
3	Script Execution and Monitoring	As a medical physicist, I want to see the output of scripts so that I can know if the scripts are executed successfully or not.	Must have	Small	• Script execution results are displayed upon completion. • Success or failure message is clearly indicated. • User can request detailed execution logs if needed.
4	Script Management and Integration	As a department administrator, I need to ensure that only authorized personnel can modify scripts so that the system integrity is maintained and misconfigurations are avoided.	Must have	Small	• Only users with specific roles can modify script settings. • Unauthorized modification attempts are logged and alerted. • System performs regular audits of script modifications.
5	Script Management and Integration	As a user(all positions), I want to easily upload new scripts into the interface while ensuring they meet operational standards so that users can access the latest scripts and the system remains secure and functional.	Must have	Small	• Users can upload scripts through a secure interface. • Each script undergoes a compatibility and security check. • Successful uploads are confirmed; issues prompt guidance for resolution.
6	Script Management and Integration	As a user(all positions), I want to easily delete existing scripts from the interface so that the system integrity is maintained and misconfigurations are avoided.	Must have	Small	• Users can delete scripts with proper authorization. • Deletion requests prompt a confirmation to prevent accidental loss. • System logs all deletion activities for audit purposes.
7	Script Execution and Monitoring	As a clinical researcher, I want the interface to adapt to various script outputs and allow me to view and analyze results directly so that I can efficiently conduct my studies without needing additional tools or feedback from the system.	Must have	Medium	• Interface dynamically adjusts to show various script outputs. • Researchers can filter and analyze results within the platform. • Supports exporting data for further analysis without external tools.
8	Script Execution and Monitoring	As a medical physicist, I want to save the output of the script so I can view the output repeatedly.	Must have	Small	• Users can save script outputs with date and time stamps. • Saved outputs are accessible for future reference. • System provides options for organizing and labeling saved results.
9	User Experience and dashboard design	As a medical physicist, I want to have a dashboard that displays the status of scripts, so that I can monitor the usage and functionality of the scripts being run.	Must have	Medium	• Dashboard displays real-time status of script executions. • Users can customize which script information is displayed.

					• System provides alerts or notifications for script events.
10	Script Management and Integration	As a software engineer, I want to integrate a continuous integration/continuous deployment (CI/CD) pipeline into our GitHub repository for testing and deployment of our scripts.	Must have	Medium	• Integration of CI/CD pipeline into GitHub repository. • Pipeline supports automatic testing and deployment of scripts. • Documentation provided for setup and management of the CI/CD process.
11	User access and Security	As a clinical researcher, I want to log in to the web interface, so that I can access, manage and execute the scripts helping analyse patients' data.	Should have	Medium	• Secure login mechanism for accessing the web interface. • Users can manage and execute scripts post-login. • System supports password recovery and account management.
12	User Experience and dashboard design	As a radiation therapist, I need to access detailed usage instructions for each script directly from the dashboard so that I can use them correctly without extensive training.	Should have	Small	• Dashboard provides access to detailed script instructions. • Instructions include prerequisites, steps, and expected outcomes. • Accessibility features ensure instructions are easy to understand.
13	Script Management and Integration	As a system administrator, I want to categorize and label additional scripts for future integration so that the system can easily scale up to 20 scripts without confusing the users.	Could have	Medium	• Admin can create and manage script categories. • Interface supports adding labels and descriptions to scripts. • System can scale to support up to 20 categorized scripts.
14	User Experience and dashboard design	As a research fellow, I want a user-friendly dashboard so that I don't need to spare extra time to learn how to use the dashboard.	Could have	Medium	• The dashboard design follows established UX/UI best practices. • Key functionalities are accessible within two clicks from the home screen. • A brief interactive tutorial is available for first-time users.
15	User Experience and dashboard design	As a medical physicist, I want to customize my dashboard to highlight frequently used scripts and critical information so that I can streamline my daily workflow and improve efficiency.	Could have	Medium	• Users can customize the layout and information on their dashboard. • Changes to the dashboard are saved and persist between sessions. • Customization options are intuitive and require minimal training.
16	User access and Security	As an IT department engineer, I want to be able to grant account access to the web interface so that security of the web interface and data is guaranteed.	Could have	Medium	• IT can grant or revoke web interface access. • Access levels are assignable based on user roles. • All changes to account access are logged for security auditing.

17	Script Management and Integration	As a software developer, I want a comprehensive deployment guide for the Flask-based web interface so that I can efficiently set it up on our servers without having to learn new technologies.	Could have	Small	• The guide provides step-by-step instructions for deploying the Flask web interface. • Documentation includes requirements for server environments and dependencies. • Troubleshooting section addresses common issues during deployment.
18	Script Management and Integration	As a software developer, I need the website to integrate smoothly with SQL-based databases so that I can manage data consistently with other applications we use.	Could have	Medium	• The web interface seamlessly connects to SQL-based databases. • Documentation details the process for setting up and configuring the database connection. • The system supports data consistency checks and conflict resolution.
19	User access and Security	As a research fellow, I want to access different users' output records so I can teamwork better.	Will not have	Medium	• Research fellows can access output records of different users based on permissions. • The interface allows filtering and searching through output records. • Collaboration tools are integrated to facilitate sharing and discussion of results.
20	Script Execution and Monitoring	As a Clinical Oncologist, I want to schedule scripts to run at specific times or intervals so that routine tasks can be automated, reducing manual effort and increasing consistency.	Will not have	Medium	• Users can schedule scripts to run at specified times or intervals. • The system provides confirmation that scripts are scheduled correctly. • Users receive notifications upon script completion or in case of errors.
21	Script Management and Integration	As a software developer, I need to implement a robust backup and recovery system for the script interface and its data so that we can quickly restore functionality in the event of a system failure or data loss.	Will not have	Medium	• The system includes options for regular automated backups. • Users can manually initiate backups and select data for backup. • The recovery process is clearly documented and can be executed with minimal steps.
22	User Experience and Dashboard Design	As a non-English speaking healthcare worker, I want the interface and documentation to be available in multiple languages so that I can understand and utilise the system effectively.	Will not have	Medium	• The interface supports multiple languages, including all UI elements and documentation. • Users can easily switch languages within the interface. • Language preferences are saved for future sessions.

23	Script Execution and Monitoring	As a software developer, I need to monitor the progress of script executions in real-time so that I can quickly address any issues that arise during the process.	Will not have	Medium	• The interface provides real-time monitoring of script executions. • Developers can set alerts for specific events or errors during execution. • The system logs all execution details for post-analysis.
24	Script Execution and Monitoring	As a software developer, I want to understand the traffic capacity and scalability of the Flask web application so that I can ensure it remains operational and responsive under expected loads.	Will not have	Medium	• Documentation outlines the web application's traffic handling capabilities. • The system includes features for monitoring and managing load. • There are clear guidelines for scaling the application in response to increased traffic.

User Stories (Sprint 2 Version)

Key content

- Summary
- User story classification
- Product backlog

According to our clients needs, product backlog is implement with epics: script execution and monitoring; script management and integration, and user access and security, and user experience and dashboard design. Product backlog is designed including prioity and size estimation, which can be considered as a requirement of project with acception criteria for each user story. The classification of prioity and size estimation is defined as following.

User stories classification:

Size Estimation	
Small	User stories that can be completed in less or equal than 1 day by one person.
Medium	User stories that can be completed in 2 to 4 days by one person.
Large	User stories that can be completed in 5 to 7 days by one person.

Priority	
Must have	Non-negotiable features that must be implemented.
Should have	Important features that can add significant value.
Could have	Features that can add value but impact a little if left out.
Will not have	Features that are not a need for the project in this specific time-frame.

Product Backlog

	Epic	As	I want to	So that	Priority	Size Estimation	Justification
1	Script Execution	Radiation therapist	Run a specific script from a web interface easily	I can perform analysis tasks using python from any location without local installations or configurations	Must have	Large	It's the initial phase of the entire project for both the frontend and backend. This stage involves time-consuming planning on how to implement script execution and ensure that long-running Python scripts can run without disconnecting.

2		Radiation therapist	Support multiple files or folders uploading	I can analysis multiple parents information simultaously	Must have	Large	Script execution, highly demanded by clients, requires the capability to upload multiple files. We will need to design a way to modify file paths, taking into account that scripts may specify particular path names.
3		Radiation therapist	Obtain the output of scripts	I can get wanted one or lots of processed pateints information	Must have	Small	Users require modifications to patient information. It is straightforward to implement generated file downloads within the existing framework.
4		IT department developer	Website to integrate smoothly with SQL-based databases	I can manage data consistently with other applications we use.	Must have	Small	The IT department of the client company has requested an easy-to-use SQL-based database, a common requirement in the technology sector.
5	Script Management	Radiation therapist	Have a user-friendly script execution dashboard	I can easily find and run the script I want	Must have	Large	It's important to provide clients with a list of scripts so they can easily operate the system.
6		Radiation therapist	Access detailed usage instructions for each script directly	I can use scripts efficiently and correctly without extensive training	Must have	Small	It's essential to provide instructions to users, particularly for those who have no programming knowledge.
7		Radiation therapist	Be able to save frequently used scripts	I can easily and conveniently execute the scripts I need	Must have	Medium	If the number of scripts increases, it becomes necessary to implement a save function for specific scripts. This will involve UI design and database integration.
8		Radian Researchers	version control all my scripts	I access history scripts and manage all scripts easily	Must have	Small	As the client specifically points out needs, it must be done. Github repo can easily achive this needs
9	User access	Department administrator	Have two groups of users, administrator and normal user	Coworkers with proficient coding knowledge can upload, edit and delete needed scripts and the other group can focus on script execution	Must have	Large	As the client has specifically outlined their needs, these must be addressed. A GitHub repository can easily fulfill these requirements.
10		Department administrator	Let different users access the corresponding system based on their role	They can access different features based on their permission.	Must have	Medium	Implementing role-based access control is essential for maintaining system security and ensuring users have the appropriate permissions to perform their tasks efficiently and securely. This setup prevents unauthorized access and optimizes

							the user experience by providing role-specific functionalities.
11		Department administrator	Have a role of administrator for user account management	The efficient management of users can be handles within teams.	Must have	Large	We need to design the UI and implement backend changes for improved user management, as specifically requested by the client.
12	Script Management	Medical physicist	Have a dashboard that displays the status of scripts	I can have orginased space to run script and see which script to be run.	Should have	Medium	The client may not run scripts simultaneously. Additionally, UI design modifications and updates to the current framework are required.
13		Radiation researcher	Categorize and label additional scripts for future integration	The system can easily scale up to 20 scripts without confusing the users	Could have	Medium	It's easy for users to search for the script they want. Need Ui design and backend search functionality support for quick search
14		Medical physicist	Customize my dashboard to highlight frequently used scripts and critical information	I can streamline my daily workflow and improve efficiency.	Could have	Large	Modularizing the UI could enhance the user experience, but it requires careful design and may significantly alter the previous layout.
15		IT Department developer	Have a comprehensive deployment guide for the Flask-based web interface	I can efficiently set it up on our servers without having to learn new technologies.	Could have	Small	Considering the website needs to be deployed on the client's server, detailed documentation should be written.
16		Clinical Oncologist	Schedule scripts to run at specific times or intervals	Routine tasks can be automated, reducing manual effort and increasing consistency.	Could have	Medium	Ranked as the 'could have' feature of medium priority is scheduling scripts to run at certain times or intervals as it drastically automates repetitive activities and increases operational efficiency, however, it is not a core functionality and may bring much value in daily workflow optimization

User Stories (Sprint 3 Version)

Key content

- Summary
- User story classification
- Product backlog

According to our clients needs, product backlog is implement with epics: script execution, script management, and user access and execution management. Product backlog is designed including prioity and size estimation, which can be considered as a requirement of project with acception criteria for each user story. The classification of prioity and size estimation is defined as following.

User stories classification:

Size Estimation	
Small	User stories that can be completed in less or equal than 1 day by one person.
Medium	User stories that can be completed in 2 to 4 days by one person.
Large	User stories that can be completed in 5 to 7 days by one person.

Priority	
Must have	Non-negotiable features that must be implemented.
Should have	Important features that can add significant value.
Could have	Features that can add value but impact a little if left out.
Will not have	Features that are not a need for the project in this specific time-frame.

Product Backlog

	Epic	As	I want to	So that	Priority	Size Estimation	Justification
1	Script Execution	Radiation therapist	Run a specific script from a web interface easily	I can perform analysis tasks using python from any location without local installations or configurations	Must have	Large	It's the initial phase of the entire project for both the frontend and backend. This stage involves time-consuming planning on how to implement script execution and ensure that long-running Python scripts can run without disconnecting.
2		Radiation therapist	Support multiple files or folders uploading	I can analysis multiple parents information simultaously	Must have	Large	Script execution, highly demanded by clients, requires the capability to upload multiple files. We will need to design a way to modify file paths,

							taking into account that scripts may specify particular path names.
3		Radiation therapist	Obtain the output of scripts	I can get wanted one or lots of processed pateints information	Must have	Small	Users require modifications to patient information. It is straightforward to implement generated file downloads within the existing framework.
4	Script Management	Radiation therapist	Have a user-friendly script execution dashboard	I can easily find and run the script I want	Must have	Large	It's important to provide clients with a list of scripts so they can easily operate the system.
5		Radiation therapist	Access detailed usage instructions for each script directly	I can use scripts efficiently and correctly without extensive training	Must have	Small	It's essential to provide instructions to users, particularly for those who have no programming knowledge.
6		Radiation therapist	Be able to save frequently used scripts	I can easily and conveniently execute the scripts I need	Must have	Medium	If the number of scripts increases, it becomes necessary to implement a save function for specific scripts. This will involve UI design and database integration.
7		Radian Researchers	version control all my scripts	I access history scripts and manage all scripts easily	Must have	Small	As the client specifically points out needs, it must be done. Github repo can easily achive this needs
8		Radiation researcher	Categorize and label additional scripts for future integration	The system can easily scale up to 20 scripts without confusing the users	Must have	Medium	It’s easy for users to search for the script they want. Need Ui design and backend search functionality support for quick search.
9	User access	Department administrator	Have two groups of users, administrator and normal user	Coworkers with proficient coding knowledge can upload, edit and delete needed scripts and the other group can focus on script execution	Must have	Large	As the client has specifically outlined their needs, these must be addressed. A GitHub repository can easily fulfill these requirements.
10		Department administrator	Let different users access the corresponding system based on their role	They can access different features based on their permission.	Must have	Medium	Implementing role-based access control is essential for maintaining system security and ensuring users have the appropriate permissions to perform their tasks efficiently and securely. This setup prevents unauthorized access and optimizes the user experience by providing role-specific functionalities.
11		Department	Have a role of administrator for user	The efficient management of users	Must have	Large	We need to design the UI and implement backend changes for

		administrator	account management	can be handles within teams.			improved user management, as specifically requested by the client.
12		Department administrator	User info containing 6 digit-username and could reset password to all	Our team member can find avoid complex and multi-step login interface.	Must have	Small	Client required. And it's easy to implement based on current architecture
13		Department administrator	User info containing full name shown	Our team member can easily find out how upload the scripts	Must have	Small	Client required. And it's easy to implement based on current architecture
14	Execution history management	Radiation researcher	Have a dashboard that displays the status of scripts	I can have orginased space to run script and see which script to be run.	Should have	Medium	The client may not run scripts simultaneously. Additionally, UI design modifications and updates to the current framework are required.
15		Radiation researcher	Access to previrous execution output	I can I can review my past operations.	Should have	Medium	This feature is important because some users may need to reuse previous outputs if they have lost the downloaded output or are working on different devices.
16		Radiation researcher	Have execution history	I can know what operation I have done.	Should have	Medium	It would be useful if user can view their execution history for reference. It requires a new section degisn for both frontend and backend.
17		Radiation researcher	Notify user when long script executions are done	I can know when my python script execution is done	Should have	Large	Some scripts' responsing timing is very long. It would be helpful, they will be notified when their long execution is done. However, this may include modification of current archtecture.
18	Handover	IT department developer	Website to integrate smoothly with SQL-based databases	I can manage data consistently with other applications we use.	Must have	Small	The IT department of the client company has requested an easy-to-use SQL-based database, a common requirement in the technology sector.
19		IT department developer	Website to integrate smoothly with SQL-based databases	I can manage data consistently with other applications we use.	Must have	Small	The IT department of the client company has requested an easy-to-use SQL-based database, a common requirement in the technology sector.

How to run scripts shared by the client

Download the latest version data shared by the client from Google Drive

Script 1 Copy Monaco Patient

1. Replace the original code with below code

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 18 08:33:17 2022

@author: MCDELZ
"""

import os, re
import shutil
import ctypes  # An included library with Python install.
from sys import exit


current_folder_path = os.getcwd()
# this is scripted, better to ask user to select 1 or multiple target plan folders & destinations to copy & cha
location1 = f"{current_folder_path}{os.sep}location1"
location2 = f"{current_folder_path}{os.sep}location2"

# list of plan folders to change, format: P[[old name1, new name1], [old name2, new name2]]
P = [ ['zzACDSU','RzzACDSUtest'] ]


# first copy folder from location 1 to location 2, then edit location 2

for pat in P:    # pat = P[0]

    IDold = pat[0]
    IDnew = pat[1]


     # 1 ---- folder name----
    path1 = location1 + os.sep + '1~' + IDold + os.sep
    path2 = location2 + os.sep + '1~' + IDnew + os.sep

    if os.path.exists(path2):
        # ctypes.windll.user32.MessageBoxW(0, "The new plan folder already exists", "Not copied", 0)
        print("The new plan folder already exists")
        exit()
    else:
        shutil.copytree(path1, path2)




    # 2 ---- demog.xxx
    path = location2 + os.sep + '1~' + IDnew + os.sep

```

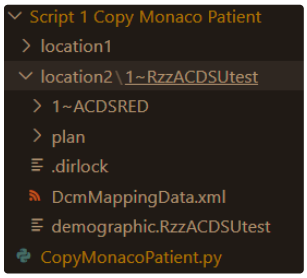
```
    old = path + 'demographic.'+ IDold
    new = path + 'demographic.'+ IDnew
    if os.path.exists(old):
        os.rename(old, new)

        with open(new, 'r') as filepointer:
            filedata = filepointer.read() #open plan file

        filedata2  = filedata.replace('\n' + IDold + '\n', '\n'+IDnew+'\n') #replace ID

        with open(new, 'w') as filepointer:
            filepointer.write(filedata2) #write new plan file



    # 3 ---- plan / xxx.hyp----

    path = location2 + os.sep + '1~' + IDnew + os.sep + 'plan'

    plist = os.listdir(path)

    for planfolder in plist:
        old = path + os.sep + planfolder + os.sep + IDold + '.hyp'

        if os.path.exists(old):
            new = path + os.sep + planfolder + os.sep + IDnew + '.hyp'
            os.rename(old, new)


        else:
            print(f"skip {old}")

        oldb = path + os.sep + planfolder + os.sep + IDold + '_rxB.hyp'
        if os.path.exists(oldb):
            newb = path2 + os.sep + planfolder + os.sep + IDnew + '_rxB.hyp'
            os.rename(oldb, newb)
        # else:

            # print(f"skip {oldb}")




    # 4 ---- plan/plan.txt > xxx----

    path = location2 + os.sep+'1~' + IDnew + os.sep + 'plan'

    plist = os.listdir(path)

    for planfolder in plist:
        planpath =   path + os.sep + planfolder + os.sep + 'plan'

        if os.path.exists(planpath):
            with open(planpath, 'r') as filepointer: filedata_old = filepointer.read()
            if filedata_old.find('\n'+IDold+'\n')>0:
                filedata_new  = filedata_old.replace('\n'+IDold+'\n', '\n'+IDnew+'\n') #replace ID
                with open(planpath, 'w') as filepointer: filepointer.write(filedata_new)
```

```
        planpathB =   path + os.sep + planfolder + os.sep + 'rxB_plan'
        if os.path.exists(planpathB):
            with open(planpathB, 'r') as filepointer: filedata_old = filepointer.read()
            if filedata_old.find('\n'+IDold+'\n')>0:
                 filedata_new  = filedata_old.replace('\n'+IDold+'\n', '\n'+IDnew+'\n') # replace ID
                 with open(planpathB, 'w') as filepointer: filepointer.write(filedata_new)









    # 5 ---- CT / info.txt > xxx----

    path = location2 + os.sep + '1~' + IDnew
    if os.path.exists(path):
        clist = [item for item in os.listdir(path) if re.match(r'[0-9]+~*', item)]

        for ct in clist:
            ctpath =   path + os.sep + ct + os.sep + 'info'

            with open(ctpath, 'r') as filepointer:
                filedata_old = filepointer.read()

            filedata_new  = filedata_old.replace('~'+IDold+'\n', '~'+IDnew+'\n') #replace ID
            filedata_new  = filedata_new.replace('\n'+IDold+'\n', '\n'+IDnew+'\n')

            with open(ctpath, 'w') as filepointer:
                filepointer.write(filedata_new)


    print(f"The new plan folder copied & changed!\n from: \n{path1}\n\nto: \n{path2}")
    # ctypes.windll.user32.MessageBoxW(0, f"The new plan folder copied & changed!\n from: \n{path1}\n\nto: \n{p
```

2. Run the script

```
python CopyMonacoPatient.py
```

3. Inspect the result in “location2” folder



Script 2 Reorganise Mim Export Data

Replace the code

```
# -*- coding: utf-8 -*-
"""
```

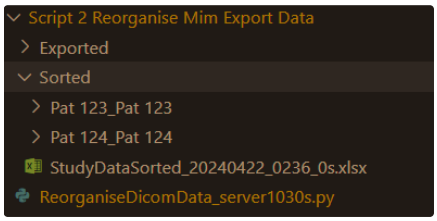
```
Created on Tue May 18 14:16:05 2021

@author: Leah & Jess
"""

# organise single folder level into ID - date_Scantype - scan name / RTPlan-Struct-Dose / other
# ignore PR, common text at start of scan name

# read main folder, for each folder get ID, name, date, scan type, scan name

# to do:
# keep 3 layers: L1 = patient, L2 = study description, L3 = series description / other
# read in export folder (month_Studies), copy to sorted folder, if patient exists, write scans & studies inside
# instead of index, create parameter table, write to excel

import os, time
import shutil
import pandas as pd
from   datetime import datetime


t1 = time.perf_counter()
def clean_path(path):
    path = path.replace('/',os.sep).replace('\\',os.sep)
    if os.sep == '\\' and '\\\\?\\' not in path:
        # fix for Windows 260 char limit
        relative_levels = len([directory for directory in path.split(os.sep) if directory == '..'])
        cwd = [directory for directory in os.getcwd().split(os.sep)] if ':' not in path else []
        path = '\\\\?\\' + os.sep.join(cwd[:len(cwd)-relative_levels]\
                        + [directory for directory in path.split(os.sep) if directory!=''][relative_levels:])
    return path

# sourcepath = r"H:\Restrict\Radiotherapy\DicomData\exported"
# sortedpath = r"H:\Restrict\\Testsort\testsortNonUnity"

# ***change to open dialogue, defaults to this one, but possible to change
sourcepath = r"\\server1030s\DataRegistry\MIM EXPORT\EXPORT Research_MR"
sortedpath = r"\\server1030s\DataRegistry\StudyDataSorted"
redcappath = r"\\server1030s\DataRegistry\StudyDataSorted\RedCap_MRRegistry_DATA_2023-04-14_0850.csv"

# root = tk.Tk()
# root.withdraw()
# sourcepath = filedialog.askopenfilename(initialdir=sourcepath)
current_folder_path = os.getcwd()
sourcepath = f"{current_folder_path}\Exported"

print('sourcepath:', sourcepath)


sortedpath = f"{current_folder_path}\Sorted"
IndexFile = sortedpath +os.sep+'StudyDataSorted'+ datetime.now().strftime("_%Y%m%d_%H%M")+".xlsx"
print('IndexFile:', IndexFile)
#*** read in RedCap file, use Mosaiq to get list of UR>Subfile, get list of IDs

os.chdir(sortedpath)

print(f'started {t1}')
```

```
sites = {   'abdo':         ['abdo', 'abdomen'],
            'abdo.upp':     ['abdomen upper','upper abdo',  'duod',  'stomach'],
            'abdo.lwr':     ['abdomen lower','lower abdo','colon'],
            'liver':        ['liver'],
            'kidney':       ['kidney'],
            'pancreas':     ['pancreas'],
            'brain':        ['brain'],
            'breast':       ['breast'],
            'chest':        ['chest'],
            'hn':           ['head','neck','HandN','HNbilat','HNipsi', 'H&N', 'H.&.N'],
            'lung':         ['lung'],
            'oesophagus':   ['oesoph','esoph'],
            'pelvis':       ['pelvis', 'pelvis general','bowel'],
            'rectum':       ['rectum'],
            'bladder':      ['bladder'], # bladder, bowel, uterus and cervix, vagina, and rectum
            'gynae':        ['gynae'],
            'prostate':     ['prostate', 'prost', 'pros'],
            'spine':        ['spine'],
            'pelvis.spine': ['pelvis&spine'],
            'bone':         ['femur']
            }

techs = {'stereo': ['stereo', 'sbrt', 'srs'],
        'pall': ['pall', 'palliative'],
        'FB.BH':['free', 'bh'],}


studymonths = os.listdir(sourcepath)
print(studymonths)

for studymonth in studymonths:
    #studymonth = studymonths[0] #test


    folders = os.listdir(sourcepath + os.sep + studymonth)
    # f = folders[2] # testing

    for f in folders:  # 0 = patname, 1 = patid, # 2 = modality, 3 = studydate, 4 = studytime 5 = studydesc, si
        # f = folders[0] #test
        HDx = {'patname':'', 'patid':'', 'modality':'', 'studydate':'', 'studytime':'',  'studydesc':'', 'serie
        for hd in list(HDx.keys()):
            HDx[hd] = f.split('_')[list(HDx.keys()).index(hd)]

        # *** check if ID is in the RedCap list, skip if not.




        usesite = 'sx'
        for sitegroup in list(sites.keys()):
            for site in sites[sitegroup]:
                if site in HDx['studydesc'].lower(): #'prostate': ['prostate', 'prost', 'pros','Prostate'],
                    usesite = sitegroup
        HDx['site'] = usesite

        usetech = 'tx'
        for techgroup in list(techs.keys()):
            for tech in techs[techgroup]:
```

```
                if tech in HDx['studydesc'].lower():
                    usetech = techgroup
        HDx['tech'] = usetech

        inst = HDx['instance']
        try: HDx['instance'] = int(inst)
        except: HDx['instance'] = inst.lstrip('0')



        #level thing:
        L1 = HDx['patid'] + '_' + HDx['patname']
        L2 = HDx['studydate'] + '_' + HDx['modality']  + '_' + HDx['site']   + '_' + HDx['tech']
        L3 = HDx['studytime'] + '_' + HDx['studydesc'] + '_' + HDx['series'] + '_' + HDx['nslices'] + '_' + str

        n = int( HDx['nslices'].split('n')[1]) # number of dicom files/slices
 #       if n<3 or len(HDx['series'])<2 or HDx['modality'] not in ['MR', 'CT', 'PR']:  L2 = 'other_'+L2 #JL cha
        if HDx['modality'] not in ['MR', 'CT', 'RTDOSE', 'RTst', 'RTPLAN', 'PT']:  L2 = 'other_'+L2

        scanfolder1 = sourcepath+ os.sep+studymonth+os.sep+f
        # print('scanfolder1:', scanfolder1)
        scanfolder2 =  sortedpath + os.sep + L1 + os.sep + L2 + os.sep + L3
        # print('scanfolder2:', scanfolder2)
        pat = HDx['patid'] + '-' + HDx['patname']  + ' '+  HDx['nslices']
        print(f'folder:{studymonths.index(studymonth)}/{len(studymonths)} study:{folders.index(f)}/{len(folders
        print(' ')

        if not os.path.exists(scanfolder2):   shutil.copytree( clean_path(scanfolder1) ,clean_path(scanfolder2)
        # else: print('skip')


t2 = time.perf_counter()
print(f"{(t2-t1)/60:0.1f}min for sorting" )

#%%

# ----------------------------------------
# create and excel file with index table
# ----------------------------------------
t3 = time.perf_counter()
HDx = ['PatientID', 'Name', 'other', 'StudyDate', 'Modality', 'Site', 'Technique',
       'StudyTime', 'StudyDescription','SeriesDescription','Nslices','Instance']

DicomList = pd.DataFrame(columns = HDx)
Flist = []

for root, rdirs, rfiles in os.walk(sortedpath):  # takes a while\

    level = root.replace(sortedpath, '').count(os.sep)

    if level == 3:
         # print(root)
         folder = root.replace(sortedpath, '').replace(os.sep, '_').split('_')
         # eg "\\server1030s\DataRegistry\StudyDataSorted\211230_RYAN^ANTHONY^FRANCIS\2021-11-18_MR_oesophagus_



         folder = folder[1:] #drop 1st empty value
         Flist.append(folder)
```

```
        ind = Flist.index(folder)
        if folder[2]!='other': folder.insert(2, '-')

        for i in range(len(HDx)):
            try: DicomList.loc[ind, HDx[i]] = folder[i]
            except: DicomList.loc[ind, HDx[i]] = '-'

#*** add Redcap data columns for each ID

t4 = time.perf_counter()

#%%

diff = t2-t1
txt = 'sort finished'
if diff<60:
    diffs = f'_{diff:0.0f}s'
    print(f"{txt}, took {diff:0.1f} seconds" )
elif diff<3600:
    diffs = f'_{diff/60:0.0f}min'
    print(f"{txt}, took {diff/60:0.1f} minutes" )
else:
    diffs = f'_{diff/3600:0.0f}h'
    print(f"{txt}, took {diff/3600:0.1f} hours" )
IndexFile = IndexFile.replace(".xlsx", f"{diffs}.xlsx")
DicomList.to_excel(IndexFile, sheet_name = 'DicomList')


diff = t4-t3
txt = 'table finished'
if diff<60:        print(f"{txt}, took {diff:0.1f} seconds" )
elif diff<3600:    print(f"{txt}, took {diff/60:0.1f} minutes" )
else:              print(f"{txt}, took {diff/3600:0.1f} hours" )
```

2. Run the script
3. Result: see “Sorted” folder



Script 3 Edit Monaco Template

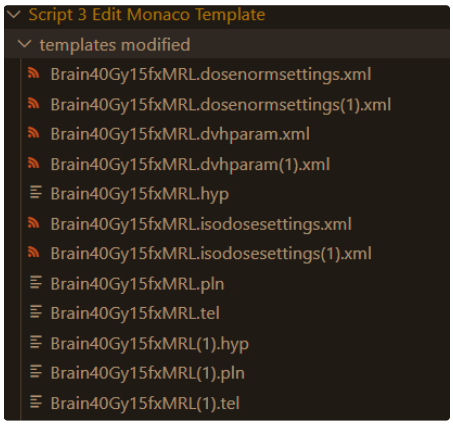
1. Replace the code

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jul 21 14:58:09 2022

@author: MCDELZ
"""

```

```
8
9
10  import os, re, time
11  from pathlib import *
12  import shutil
13
14
15  #list 00 files
16  # folderpath = "\\\\cmsada2\\FocalData\\MonacoTemplates"
17  # files00 = [f.split('.tel')[0] for f in os.listdir(folderpath) if f.endswith('.tel')]
18  # files00 = [f for f in files00 if f.startswith('00')]
19  t1 = time.perf_counter()
20
21  current_folder_path = os.getcwd()
22  editfolder = f"{current_folder_path}{os.sep}templates original"
23  donefolder = f"{current_folder_path}{os.sep}templates modified"
24
25  # rename
26  files = os.listdir(editfolder)
27
28  files = [f for f in files if os.path.isfile(editfolder + os.sep + f)] # remove folders
29
30  for f in files:
31      f2 = re.sub('New00', '00', f)
32
33      ind = f2.find('v0')
34      if ind>=0:
35          vsn = str(int(f2.split('v0')[1][0])+1)
36          f3 = re.sub(r'v0\d',  r'v0'+vsn, f2)
37      else:
38          f3 = f2
39      shutil.copy(editfolder + os.sep + f, donefolder + os.sep + f3) # write to done folder
40
41
42  t2 = time.perf_counter()
43  s = 'remove new, add 1 to v'
44  print(s + " (" + f"{t2-t1:0.1f}s)" )
45  t1 = time.perf_counter()
46
47
48
49
50  # remove :
51  # source = clinicfolder
52  source = donefolder
53  files = os.listdir(source)
54  files = [f for f in files if os.path.isfile(source + os.sep + f)] # remove folders
55  files = [f for f in os.listdir(source) if f.endswith('.tel')]
56
57  colonset = []
58
59  for f in files:
60      if f.endswith('.tel'):
61
62          telpath = source + os.sep + f
63
64          with open(telpath, 'r') as filepointer:   filedata = filepointer.read() #open plan file
65
```

```
        indset = [i for i in range(len(filedata)) if filedata.startswith(':',i)]

        colonset.append([f, indset])

        if len(indset)==2:
            for i in range(2):
                ind1 = indset[i] - filedata[indset[i]-3:indset[i]].find('\n')
                ind2 = indset[i] + filedata[indset[i]:indset[i]+50].find('\n')

                line1 = filedata[ind1:ind2]

                print(f"{f} \n{line1}") # show lines with :

            #edit & overwrite tel file

            filedata2  = filedata.replace(': ', ' ') #replace :
            filedata2  = filedata.replace(':', ' ') #replace :

            with open(telpath, 'w') as filepointer:  filepointer.write(filedata2) #write new plan file

t2 = time.perf_counter()
s = 'remove : from tel'
print( s + " (" + f"{t2-t1:0.1f}s)" )
t1 = time.perf_counter()
```

2. Run the script
3. Inspect the result
4. (templates modified)



Project Management

We utlize Trello as task management tool, confluence as documentation reference, Github as code collaboration platform, ApiPost as frontend and backend api connection platform.

Trello Workflow

Owned by Lingyi Kong • Updated on May 23, 2024

Trello Structure Sprint 2 and Sprint 3's works are tracked in Trello. Each sprint corresponds to four lists, which are 'Sprint Backlog', 'Task - To Do', 'Task - In Progress', and 'Task - Complete'. Trello Workflow User stories are initially placed in the 'Sprint Backlog' list at the beginning of each sprint. Once finishing task assignment, more det

Confluence

Open preview

Github Workflow

Owned by Chloe_Duan • Updated on May 23, 2024

Our development workflow is centered around three main types of branches: main , dev , and feature branches. Each branch serves a specific purpose in the lifecycle of our application development, ensuring a streamlined and organized process. Branches Main Branch : The main branch is our production branch. This branch contains the project's release

Confluence

Open preview

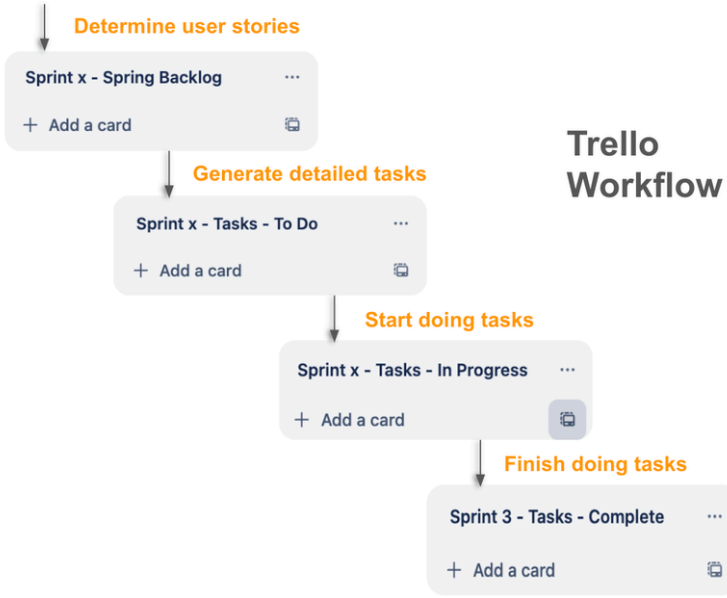
Trello Workflow

Trello Structure

Sprint 2 and Sprint 3's works are tracked in Trello. Each sprint corresponds to four lists, which are 'Sprint Backlog', 'Task - To Do', 'Task - In Progress', and 'Task - Complete'.

Trello Workflow

User stories are initially placed in the 'Sprint Backlog' list at the beginning of each sprint. Once finishing task assignment, more detailed and particular actions, as we say the tasks, will be placed in 'Task - To Do' list. Before starting a task, team members are moved to move the specific task card to 'Task - In Progress' list. As the time a task is done, team members can eventually move the task card to 'Task - Complete' list.



Card Number Explanation

We use an extension in Trello to give every card a unique ID in the order of card creation time. Both user stories and tasks are assigned unique numbers automatically when the cards are created. To make connections between user stories and their associated tasks, we manually add a serial number at the beginning of each task.

This picture shows an example of how a task is associate with an user story. The task assigned to #95 belongs to the user story assigned to #78. ‘PBI’ refers to ‘Product Backlog Identifier’.


Sprint 2 - Sprint Backlog
Must have Large
As a Radiation therapist, I want to have a user-friendly script execution dashboard so that I can easily find and run the script I want.
#77
Must have Small
As a Radiation therapist, I want to access detailed usage instructions for each script directly so that I can
ciently and correctly
ive training.
Unique ID
1 #78
Must have Medium
As a Radiation therapist, I want to be able to save frequently used scripts so that I can easily and conveniently execute the scripts I need.
1 #79
Sprint 2 - Tasks - To Do
Add a card
Sprint 2 - Task - In progress
Add a card
Link from task to user story
Sprint 2 - Tasks -Complete
Backend Task
PBI-78 add script api
#90
Backend Task
PBI-78 Delete scripts api
#93
Serial Number
Backend Task
PBI-78 Update script api
#95
Backend Task
PBI-78 List script api
#96
Backend Task
PBI-78 Get script api
#98
Backend
PBI-76 Design database schema

Github Workflow

Our development workflow is centered around three main types of branches: `main`, `dev`, and feature branches. Each branch serves a specific purpose in the lifecycle of our application development, ensuring a streamlined and organized process.

Branches

- **Main Branch**: The `main` branch is our production branch. This branch contains the project's release history and is always in a deployable state.
- **Dev Branch**: The `dev` branch serves as the integration branch for features. It is the default branch where all feature branches are merged and tested together. Once the team is confident in the stability of `dev`, it is merged into `main` for release.
- **Feature Branches**: Feature branches are used for developing new features or bug fixes. They are created from the `dev` branch and should be named according to the conventions outlined below to reflect their purpose and the technology stack they impact.

Naming Conventions

To maintain clarity and consistency, we follow specific naming conventions for our feature branches. These conventions help in identifying the purpose of the branch, the technology stack involved, and the feature or bug it addresses.

General Format: `<type>/<tech>-<description>`

- `<type>`: Indicates the purpose of the branch (e.g., `feat`, `fix`, `refactor`, `docs`, etc.).
- `<tech>`: Specifies the technology stack or component the branch focuses on (`react` for frontend changes, `py` for backend changes).
- `<description>`: A brief, hyphen-separated description of the feature or fix.

Code Review

We utilize an AI code review provided by the teaching team, which automatically checks code quality with every pull request. This ensures that our codebase remains clean and maintainable.

Meeting notes

Client meeting：

Date	Meeting notes
Mar 15, 2024	2024-03-15 Client meeting notes
Apr 10, 2024	2024-04-10 Client meeting notes
May 3, 2024	2024-05-03 Client meeting notes
May 16, 2024	2024-05-16 Client Meeting notes
Jun 5, 2024	2024-06-05 Client Meeting notes

Supervisor Standup meeting：

Date	Meeting notes
Mar 20, 2024	2024-03-20 Stand up meeting with Wei
Mar 24, 2024	2024-03-24 Sprint 1 Review
Apr 10, 2024	2024-04-10 Stand up meeting with Wei
Apr 17, 2024	2024-04-17 Stand up meeting with Mingye
Apr 24, 2024	2024-04-24 Stand up meeting with Mingye
May 1, 2024	2024-05-01 Stand up meeting with Wei
May 9, 2024	2024-05-09 Stand up meeting with Wei
May 22, 2024	2024-05-22 Pre-meeting with Wei

Inner team meeting：

Date	Meeting notes
Mar 18, 2024	Sprint 2 Daily Stand-up Meeting
Mar 21, 2024	Sprint 3 Daily Stand-up Meeting
Jun 2, 2024	Sprint 4 Planning Meeting

Client Meetings

Create meeting note

Incomplete tasks from meetings

Task report

Looking good, no incomplete tasks.

Decisions from meetings

Page Title	Decisions
2024-05-03 Client meeting notes	Update user stories according to client feedback Discuss how to handle script running iterately
2024-05-16 Client Meeting notes	No need to change current path No need to add folder iteration handling in web server

All meeting notes

Title	Creator	Modified
2024-06-05 Client Meeting notes	Chloe_Duan	20 minutes ago
2024-05-16 Client Meeting notes	Chloe_Duan	Jun 01, 2024
2024-05-03 Client meeting notes	Chloe_Duan	May 24, 2024
2024-04-10 Client meeting notes	Zhihao Liang	May 24, 2024
2024-03-15 Client meeting notes	Zhihao Liang	May 24, 2024

2024-06-05 Client Meeting notes

🗓️ Date

Jun 5, 2024

👥 Participants

- @Chloe_Duan
- @Zhihao Liang
- @Yuncong Ji
- @Junhao KONG
- Client

🥅 Goals

- Communicate with client about handover section
- Record feedback video for product demo

🗣️ Discussion topics

Time	Item	Notes
June 5	Ask for feedback	• Editing to video
June 5	Handover	• Detailed deployment guidance to IT department • Zip code file emailing to client

✅ Action items

- [x] Write detailed deployment documentation to client @Zhihao Liang DUE: Jun 5, 2024
- [x] Any concerns that need client to bring up to their IT department DUE: Jun 5, 2024
- [x] Email client with packaged zip file @Chloe_Duan DUE Jun 7, 2024

2024-05-16 Client Meeting notes

🗓️ Date

May 16, 2024

👥 Participants

- Clients
- @Chloe_Duan
- @Zhihao Liang
- @Lingyi Kong
- @Yuncong Ji
- @yijun liu
- @Junhao KONG

🥅 Goals

- Ask client to define how exactly they want to utilize input folder and subfolders to iretate scripts (Better to clarify in scenrios, e.g. using Script 2)
- Ask client to clarify how they want to choose input path

🗣️ Discussion topics

Item	Presenter	Notes
Goals 1	@Chloe_Duan @Zhihao Liang	• We were afraid learning how to write proper for-loop for their python script can be challenging to client. • Client don't feel chanlleging to do so
Goals 2	@Chloe_Duan @Zhihao Liang	• Client approvaled our current deisgn of choosing input path
Any feedback from client?	@Chloe_Duan	No further improvement needed

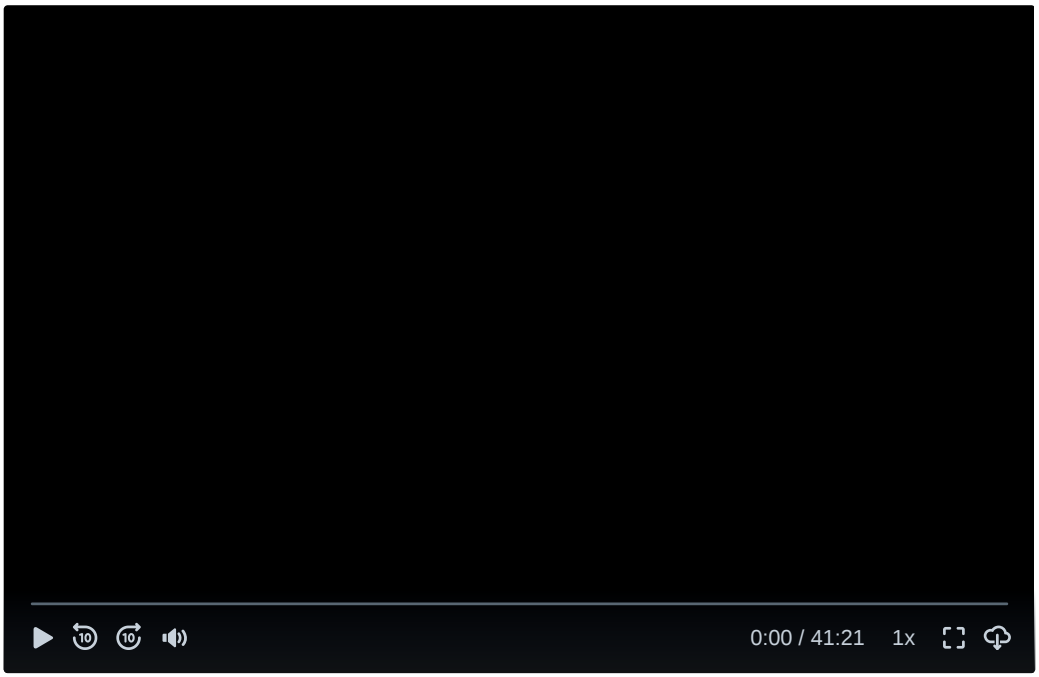
✅ Action items

- [x] Assign current user stories and due date to team member @Chloe_Duan @Zhihao Liang DUE: May 16, 2024

⤴️ Decisions

- No need to change current path
- No need to add folder iteration handling in web server

Meeting recording



2024-05-03 Client meeting notes

🗓️ Date

May 3, 2024

👥 Participants

- Client, Leah
- @Chloe_Duan
- @Zhihao Liang
- @Yuncong Ji
- @yijun liu
- @Lingyi Kong
- @Junhao KONG

🥅 Goals

- Ask client to give feedback for our sprint 2 product

🗣️ Discussion topics

Item	Presenter	Notes

Client's feedback	@Chloe_Duan	**Log-in:**
		• No email input required
		• Input 6-digit username
		• Password: set all users as the same
		User Info:
		• Username: 6 digits (e.g. LDDUAN)
		• User full name: Lin Duan
		• Password:
		  ◦ Retain the current password setup functionality
		  ◦ Initialize every user's password as the same
		  ◦ Add a button to reset everyone's password to the same value
		Additional:
		• Script => show creator's full name
		Script Execution:
		• Run script iterately

✅ Action items

- [x] Document client feedback @Chloe_Duan DUE: May 6, 2024
- [x] Update sprint 3 user stories @Chloe_Duan DUE: May 10, 2024

- [x] Team continue code development and API modifications @all DUE: May 12, 2024

Decisions

- Update user stories according to client feedback
- Discuss how to handle script running iterately

Meeting recording

2024-04-10 Client meeting notes

Meeting overview

- Meeting goals: Determine important issues such as script path and user permissions with the client.
- Key attendees: Client and all team members.
- Processes: We discussed the problems with code implementation and showed the client our initial Figma UI design for the web, which they were very pleased with and suggested areas for improvement.

Meeting minutes

Date	Attendees	Notes, decisions and action items
Apr 10, 2024	@Zhihao Liang @Lingyi Kong @Chloe_Duan @Yuncong Ji @yijun liu @Junhao KONG	1. Based on the client's satisfaction with the UI design, we decided to start implementing the front end of the web using React based on the design of Figma. 2. We decided that the team responsible for the front-end code was: @Chloe_Duan @yijun liu @Junhao KONG and the back-end team should have: @Yuncong Ji @Lingyi Kong @Zhihao Liang 3. We decided to separate the front and back ends, and the team members of the two sections would have regular meetings to communicate.

Open action items

- [x] The front and back end team members meet regularly or zoom meetings respectively. @all DUE: Mar 12, 2024
- [x] Confluence and Trello provided regular updates on our progress. @all DUE: Mar 12, 2024

Meeting recording

2024-03-15 Client meeting notes

Meeting overview

- Meet client for fitst time. Introduce team. Get project context and client needs.
- Key attendees: clients.
- Processes: Introduce with each other; Clients explain their needs; Q&A.

Meeting minutes

Date	Attendees	Notes, decisions and action items
Mar 15, 2024	All	Introduction: • Leah is looking to create a webpage to run scripts that automate various tasks in the hospital radiotherapy department • Many scripts have been created by different people to save time by automating manual processes • Goal is to have a webpage where staff can easily run scripts, even if they are not familiar with coding • If script doesn't work, physicist can look at code, fix it, and update on the webpage Webpage Requirements: • Create a webpage using a framework like Flask (open to other suggestions) • Nice design with categories/groups for scripts • Scripts will require various user inputs and produce different outputs • Need to handle running scripts in the background • Hosted on a university server initially, later deployed to hospital servers • Low expected traffic, a couple users per day Additional Tasks: • Alongside the main webpage and repository, individual team members can choose additional small scripting tasks to automate other radiotherapy processes • Examples: Averaging 3 scanned films, offline checking of patient treatment margins • Leah to provide input data, expected output, and context for each task • Scripts work with relatively small file sizes, at most a couple GB at a time Other Notes:

		• Complete freedom given on webpage design, just keep it simple and not too busy • Documentation needed at 3 levels: 1. For end users on how to use each script (content from Leah) 2. How the GitHub repo and webpage is set up 3. Code comments in modified scripts • Some security required to limit script editing to approved list of staff • No need to anonymize patient data, Leah will provide pre-anonymized sample data • No major speed requirements for most scripts, open to optimizing a couple slow ones Next Steps: • Leah to cleanup requirements doc specifying must-haves vs nice-to-haves

Open action items

- [x] Team to setup Google Drive for document sharing @all DUE: Mar 18, 2024
- [x] Schedule recurring fortnightly meetings, with Leah mainly coordinating with Kelvin and Lynn @all DUE: Mar 18, 2024
- [x] Team members to discuss splitting up tasks based on webpage, repository, scripting preferences @all DUE: Mar 18, 2024

Meeting recording

Supervisor Stand-up Meetings

Date	Meeting notes
Mar 20, 2024	2024-03-20 Stand up meeting with Wei
Mar 24, 2024	2024-03-24 Sprint 1 Review
Apr 10, 2024	2024-04-10 Stand up meeting with Wei
Apr 17, 2024	2024-04-17 Stand up meeting with Mingye
Apr 24, 2024	2024-04-24 Stand up meeting with Mingye
May 1, 2024	2024-05-01 Stand up meeting with Wei
May 9, 2024	2024-05-09 Stand up meeting with Wei
May 22, 2024	2024-05-22 Pre-meeting with Wei

2024-05-22 Pre-meeting with Wei

Meeting overview

- Meeting goals: Review final presentation slides
- Key attendees: Zhihao Liang, Lin Duan and Wei.
- Processes: We reviewed the final presentation slides.

Meeting minutes

Date	Attendees	Notes, decisions and action items
May 9, 2024	@Zhihao Liang @Wei Wang @Chloe_Duan	**Meeting Notes** • Further improvement part should be removed • Product description image should be related to product • Add refelction section to show our endeavor **Decisions** • Revise and update presentation slides. • Change presentation format from recording to live demo

Open action items

- [x] **Update slides** @Chloe_Duan DUE May 22, 2024
- [x] Further improvement part should be removed @Zhihao Liang DUE May 22, 2024
- [x] Product description image should be related to product @Chloe_Duan DUE May 22, 2024
- [x] Add refelction section to show our endeavor @Zhihao Liang DUE May 22, 2024
- [x] Change presentation format from recording to live demo @Chloe_Duan DUE May 22, 2024

2024-05-09 Stand up meeting with Wei

Meeting overview

- Meeting goals: Review current progress and communicate the problems we encounter.
- Key attendees: All team members and Wei.
- Processes: We showed Wei our current web code progress, and Wei said that she was very happy with the current progress and helped us answer questions about Confluence records.

📝 Meeting minutes

Date	Attendees	Notes, decisions and action items
May 9, 2024	@yijun liu @Zhihao Liang @Lingyi Kong @Yuncong Ji @Junhao KONG @Wei Wang	**Meeting Notes** • Discussion on the final presentation • Discussion about the combination between the sprint3 product backlog and feedback from the client. • Discussion on the way to do final presentation **Decisions** • Revise sprint 3 planning.

✅ Open action items

- [x] **Revise sprint 3 planning**

2024-05-01 Stand up meeting with Wei

Meeting overview

- Meeting goals: Review current progress and communicate the problems we encounter.
- Key attendees: All team members and Wei.
- Processes: We showed Wei our current web code progress, and Wei said that she was very happy with the current progress and helped us answer questions about Confluence records.

Meeting minutes

Date	Attendees	Notes, decisions and action items
May 1, 2024	@yijun liu @Zhihao Liang @Lingyi Kong @Chloe_Duan @Yuncong Ji @Junhao KONG @Wei Wang	**Meeting Notes** • Discussion around the challenges faced by the team in completing tasks and managing a large backlog of user stories. Concerns were raised about the level of detail in the user stories, which could be hindering effective project management. • There were discussions about reassigning tasks to different team members based on their skills and contribution levels, particularly for those struggling with the technology stack being used, such as React. • Feedback on the project's progress and demonstrations of the software were discussed, with the client's next review anticipated eagerly. • Concerns were raised about the naming and structuring of user stories, the size of user stories, and their justifications. • Discussion on the documentation and communication within the team, emphasizing the need to improve these areas to ensure all team members are actively contributing and aware of their responsibilities. **Decisions** • User stories will be consolidated to reduce the total count and improve manageability. The target is to reduce them to under 20. • Efforts will be made to involve less active team members in other aspects of the project such as documentation and client communications to ensure balanced contributions. • The team agreed on the importance of demonstrating a functional demo to the client soon and ensuring the product is ready for the next client review.

✅ Open action items

- [x] **Consolidate User Stories:** Reduce the number of user stories by grouping related items and reclassifying some as tasks within larger stories. @Chloe_Duan DUE: May 4, 2024
- [x] **Reassign Tasks:** Adjust task assignments to match team members' skills and capacities. Provide support and alternate tasks to those struggling with the current technology stack. @Zhihao Liang DUE: May 5, 2024
- [x] **Prepare for Client Demo:** Ensure the product is polished and functional for the upcoming client review. Schedule a meeting with the client to get feedback. @yijun liu DUE: May 4, 2024
- [x] **Documentation and Communication:** Improve internal documentation and communication practices. Ensure that meeting notes and decisions are clearly documented in Confluence. @Lingyi Kong DUE: May 4, 2024
- [x] **Feedback Implementation:** Implement the feedback regarding the structuring and justification of user stories in the product backlog. @Junhao KONG DUE: May 5, 2024
- [x] **Peer Review and Evaluation:** Conduct a peer review to assess individual contributions and adjust project marks accordingly. @ ALL DUE: May 6, 2024

2024-04-24 Stand up meeting with Mingye

Meeting overview

- Meeting goals: Improve the Confluence page content.
- Key attendees: All team members and Mingye Li
- Processes: We discussed the current project progress and where Confluence needs to change.

📝 Meeting minutes

Date	Attendees	Notes, decisions and action items
Apr 24, 2024	@Zhihao Liang @yijun liu @Lingyi Kong @Chloe_Duan @Yuncong Ji @Junhao KONG	**Meeting Notes** • Discussion about the project's current status including challenges with integration of the front end and back end, setup and management of GitHub branches, and code review practices. • Detailed discussions on testing strategies, including acceptance and integration testing, with a focus on improving documentation with screenshots and tester details for accountability. • Clarifications provided on project scope, necessary diagrams for documentation, and proper utilization of Confluence for organizing project information. • Advice on job hunting and improving employability with insights into the IT job market and the importance of internships and certifications. **Decisions** • Decision to continue development locally, with the client responsible for deployment on their hospital server. Clarifications from the client will be sought regarding the specifics of the deployment environment. • Emphasis on documenting the scope within the project's lifecycle, especially what is within the scope for the upcoming sprints and what isn't. • Commitment to enhancing UI design incrementally, ensuring each component is completed and tested thoroughly before moving to the next. • Approval to maintain current GitHub branch structure for development convenience but to simplify it before the project's conclusion.

✅ Open action items

- [x] **API**: connection issues resolved by @Chloe_Duan @Zhihao Liang DUE: Apr 27, 2024

- [x] **Login page**: @Chloe_Duan ; Script List and Overview page @yijun liu DUE: Apr 28, 2024 ; User Management and Script Execution page @Junhao KONG DUE: Apr 28, 2024
- [x] **Clarify Deployment Details:** Directly inquire with the client about the specifics of the server where the project will be deployed to ensure compatibility and proper setup in project documentation. @Zhihao Liang DUE: Apr 28, 2024
- [x] **Documentation Improvement:** Ensure all test cases include detailed information, such as screenshots and tester details. Incorporate a proper structure in Confluence for documenting code reviews and project scope. @Zhihao Liang DUE: Apr 28, 2024
- [x] **Diagram Inclusion:** Include necessary diagrams like ER diagrams for database structure and sequential diagrams for use cases to clarify project architecture and workflow. @Chloe_Duan DUE: Apr 27, 2024
- [x] **Code Review Documentation:** Set up a Confluence page dedicated to documenting the code review process, including a checklist and responsibilities. @Yuncong Ji DUE: Apr 28, 2024
- [x] **Prepare for Finalization:** Plan to simplify the GitHub repository structure towards the project's end, merging significant changes into fewer branches and preparing for a clean deployment. @Lingyi Kong DUE: Apr 28, 2024

2024-04-17 Stand up meeting with Mingye

Meeting overview

- Meeting Goal: Check the current process and sprint 2 progress.
- Key attendees: Mingye Li, all team members.
- Processes: Meet the new supervisor Mingye Li for first time, and he advised us on our current process. We discussed about formatting issues with Confluence and Trello, as well as issues with user stories.

Meeting minutes

Date	Attendees	Notes, decisions and action items
Apr 17, 2024	@Chloe_Duan @Lingyi Kong @Zhihao Liang @yijun liu @Junhao KONG @Yuncong Ji	• Discussion on the structure and handling of user stories, backlogs, and testing methodologies. • Considerations for API security, specifically around using session cookies or JWT for authentication. • Feedback on team contributions, emphasizing the need for improvement in areas such as documentation, test case development, and the use of Confluence for organizing content conceptually rather than sprint-based. • Advice on UI design principles and the importance of error handling and user feedback mechanisms in the application. • Suggestions on how to improve the GitHub repository with detailed change logs and handover instructions. **Decisions** • Consolidation and reclassification of user stories to ensure clarity and manageability within the project management tools. • Agreement on using JWT for session management, emphasizing the simplicity and efficiency for the current project needs. • Enhancement of UI design following recognized usability principles to ensure the application is intuitive and efficient for users.

Action items highlight

- [x] Frontend code @yijun liu @Chloe_Duan @Junhao KONG DUE: Apr 20, 2024
- [x] Backend code @Lingyi Kong @Zhihao Liang @Yuncong Ji DUE: Apr 20, 2024
- [x] **Subsequent integration:** and modification of front-end code @Chloe_Duan @Zhihao Liang DUE: Apr 21, 2024

- [x] **Enhance UI Design:** Apply the ten usability principles discussed to improve the user interface design, ensuring consistency and efficiency across the application. @yijun liu DUE: Apr 20, 2024
- [x] **Improve GitHub Documentation:** Update the GitHub repository to include a change log and detailed setup and handover instructions to assist future developers or the client in understanding and using the project. @Yuncong Ji DUE: Apr 20, 2024
- [x] **Confluence Documentation Strategy:** Restructure Confluence documentation to be concept-based rather than sprint-based to enhance the organization and accessibility of project information. @All DUE: Apr 22, 2024
- [x] **Session Management Implementation:** Implement JWT for session management and ensure proper integration and security testing. @Lingyi Kong DUE: Apr 20, 2024
- [x] **Prepare for Client Meetings:** Ensure readiness for upcoming client meetings by preparing demonstrations that showcase the progress and functionality of the application, particularly focusing on robustness and user experience. @ All DUE: Apr 22, 2024

2024-04-10 Stand up meeting with Wei

Meeting overview

- Meeting goals: Ask questions and discuss functions that the team is unsure about.
- Key attendees: All team members and Wei.
- Processes: The development of script running and user management functions is discussed.

Meeting minutes

Date	Attendees	Notes, decisions and action items
Apr 10, 2024	@Zhihao Liang @yijun liu @Lingyi Kong @Chloe_Duan @Yuncong Ji @Junhao KONG @Wei Wang	1. Split the task for Sprint 2 into different team members. 2. Organise materials on Confluence. 3. The front end team started learning the react framework and started trying to write code. 4. The back-end team started building the database and experimenting with the back-end code.

Open action items

- [x] The front-end team configured the code environment. @yijun liu @Chloe_Duan @Junhao KONG DUE: Apr 7, 2024
- [x] The back-end team set up the database. @Zhihao Liang @Yuncong Ji @Lingyi Kong DUE: Apr 7, 2024
- [x] Team members to discuss splitting up tasks based on webpage. @all DUE: Apr 9, 2024

2024-03-20 Stand up meeting with Wei

Meeting overview

- Meeting goal: Update each other's work; get feedback from supervisor.
- Key attendees: All team member.
- Processes: Stand-up meeting.

Meeting minutes

Date	Attendees	Notes, decisions and action items
Mar 20, 2024	@Zhihao Liang @Lingyi Kong @Chloe_Duan @Yuncong Ji @Junhao KONG @yijun liu @Wei Wang	1. Each member update their progress a. @Chloe_Duan and @Yuncong Ji are still working on requirement analysis b. @Zhihao Liang has initialize github repo and make a meeting note for Mar 18, 2024 meeting c. @Junhao KONG and @yijun liu showed the progress of background description and client goal and motivation and wei suggested that those part need to be revised according to workshop material. d. @Lingyi Kong is working on project backlog review. 2. Advice from Wei: a. Progress Transparency b. PO should allocate some responsibility to other team members c. Keep updating progress

Action items highlight

- [x] Allocate responsibity to other member @Chloe_Duan

Inner Team Meeting

Create from template

- Sprint 1 Daily Stand-up Meeting
- Sprint 2 Daily Stand-up Meeting
- Sprint 3 Daily Stand-up Meeting
- Sprint 4 Planning Meeting

In this section, all meetings within team members only are documented for further reference.

Sprint 1 Daily Stand-up Meeting

This is inner team meeting note in sprint 1. The shortcut is as follows.

2024-03-18 Internal meeting notes

2024-03-21 Inner-Group Assignment Discussion

2024-03-24 Sprint 1 Review

2024-03-18 Internal meeting notes

Meeting overview

- Meeting goal: Internal feedback after clients' meeting; Decide what to do next.
- Key attendees: All team member.
- Processes: Discussion; brainstorm.

Meeting minutes

Date	Attendees	Notes, decisions and action items
Mar 18, 2024	@Zhihao Liang @Lingyi Kong @Chloe_Duan @Yuncong Ji @Junhao KONG @yijun liu	1. Discuss assignment 1 2. Select a user story template 3. brainstorm user cases 4. Finalize the product backlog following the client meeting 5. Create a Trello workspace 6. Distribute Assignment 1 tasks among team members

Action Notes highlight

- [x] Background description, client goals, motivation @Junhao KONG @yijun liu DUE: Mar 23, 2024
- [x] Analysis of requirements (User Stories or Use Cases) & Plan @Yuncong Ji @Chloe_Duan DUE: Mar 24, 2024
- [x] Development environment @Lingyi Kong DUE: Mar 23, 2024
- [x] Meetings & GitHub @Zhihao Liang DUE: Mar 25, 2024

Decision

Product backblog template

2024-03-21 Inner-Group Assignment Discussion

Meeting

- Meeting goal: Check each other’s work; solve problems.
- Key attendees: All team member.
- Processes: Discussion; brainstorm.

Meeting minutes

Date	Attendees	Notes, decisions and action items
2024-03-21	@Zhihao Liang @Lingyi Kong @Chloe_Duan @Yuncong Ji @Junhao KONG @yijun liu	1. Select tool and template for Persona and Do/Be/Feel list. 2. Cooperate on creating Personas, Do/Be/Feel list and Goal Model. 3. Brainstorm on requirements based on project summary. 4. Discuss on possible implementation methods of some requirements. 5. Set a time for the next meeting.

Action items highlight

- [x] Finish user story @Chloe_Duan @Yuncong Ji DUE: Apr 25, 2024

Decision

Using Miro template for Persona and DO-BE-FEEL working board

2024-03-24 Sprint 1 Review

Meeting overview

- Meeting goal: Finalize sprint 1 content.
- Key attendees: All team members.
- Processes: Discussion about product backlog; overview whole content.

Meeting minutes

Date	Attendees	Notes, decisions and action items
2024-03-24	@Zhihao Liang @Lingyi Kong @Chloe_Duan @Yuncong Ji @Junhao KONG @yijun liu	1. Continue working on requirements (user stories), and assign priority and size estimation to each requirements. 2. Create product backlogs on Trello and distribute backlogs to Sprint 2 and Sprint 3 according to priority and estimated story points. 3. Organise materials on Confluence. 4. Tick completed items on Checklist. 5. Structure the Github repository, update Readme file, and generate Release tag. 6. Final check and ready to submit the assignment.

Action items highlight

- [x] Export PDF from confluence. @Chloe_Duan
- [x] Upload PDF to Github. @Zhihao Liang

Decision

- Go through each user story and decide for priority and size estimation
- Spilt task for sprint 2 and sprint 3

Sprint 2 Daily Stand-up Meeting

ℹ️ All team members should provide their priorities, progress, and problems twice a week in this report.

Team name	FL-Koala
Direct supervisor	@Chloe_Duan
Table of contents	

🗓️ Friday <23 April>

	Name	Priorities 🧐	Progress 😊	Problems 😐
1	@Chloe_Duan	1. Learning React and Typescript 2. Implement login page	Completed	1. Had issues with solutions of authorization 2. No idea how to connect backend APIs
2	@yijun liu	1. Learning React and Typescript 2. Finish script list page	Completed	1. Confused about JSON data format from the server. 2. Had issues with running the backend project
3	@Junhao KONG	1. Learning React and Typescript 2. Code for User management page	Completed	1. Not easy-to-understand template code 2. Had issues with connecting the user database
4	@Yuncong Ji	1. Learning Flask and Python 2. Design database schema and associated apis 3. Finish script module in Flask	Completed	1. Had difficulty in running the backend project 2. Database integration challenges
5	@Zhihao Liang	1. Implement authorization module	Completed	No

		2. Set up cloud database		
6	@Lingyi Kong	1. Code for Favorite backend 2. Design APIs 3. Learning Flask and Python	Completed	No

🗓️ Thursday <25 April>

	Name	Priorities 🧐	Progress 😊	Problems 😐
1	@Chloe_Duan	1. Debug broken issue 2. Script List api connect 3. integration three page 4. forget password	Completed	1. Inactive backend team members 2. Data format confusion about script list 3. URL routing in Next.js
2	@yijun liu	1. Dashboard UI design 2. Script list filter 3. Script list running button	Completed	1. Responsive design compatibility issues
3	@Junhao KONG	1. Script running page 2. Fix layout errors	Completed	1. User information loss 2. Steep learning curve about next.js
4	@Yuncong Ji	1. UML image generation 2. Refactor previous code	Completed	No
5	@Zhihao Liang	Code for user management API	Completed	No
6	@Lingyi Kong	Code for authentication backend	Completed	No

🗓️ Tuesday <27 April>

	Name	Priorities 🧐	Progress 😊	Problems 😐
1	@Chloe_Duan	Connect the APIs for frontend and backend	Completed	1. Incomplete input from some team members
2	@yijun liu	Code for script overview page	Completed	1. Configuration issues with

				deployment scripts
3	@Junhao KONG	Code for script execution page	Completed	1. Complexity in handling user permission
4	@Yuncong Ji	Code for script backend and database	Completed	No
5	@Zhihao Liang	Code for script execution	Completed	No
6	@Lingyi Kong	Code for authentication backend	Completed	No

🗓️ Monday <30 April>

	Name	Priorities 🧐	Progress 😊	Problems 😐
1	@Chloe_Duan	1. Connect the frontend and backend APIs 2. Merge frontend branches	Completed	1. Confused about setState in Reactjs 2. Had issues with inserting image
2	@yijun liu	Edit Confluence meeting minutes and code review as well as test case	Completed	No
3	@Junhao KONG	Edit Confluence code review and test cases	Completed	No
4	@Yuncong Ji	Code for backend and work for confluence, website test	Completed	No
5	@Zhihao Liang	Code for refactoring existing backend code	Completed	No
6	@Lingyi Kong	Edit Confluence retrospectives, code review and test cases	Completed	No

Sprint 3 Daily Stand-up Meeting

All team members should provide their priorities, progress, and problems twice a week in this report.

Team name	FL-Koala
Direct supervisor	@Chloe_Duan
Table of contents	

🗓️ Friday <01 May>

	Name	Priorities 🧐	Progress 😊	Problems 😐
1	@Chloe_Duan	1. Update login feature based on client's feedback	Completed	No
2	@yijun liu	1. Learning React and Typescript	Completed	No
3	@Junhao KONG	1. Sidebar re-design	Completed	No
4	@Yuncong Ji	1. Explore big file uploading solution	Completed	No
5	@Zhihao Liang	1. Implement listen tasks module	Completed	No
6	@Lingyi Kong	1. Implement Executions API	Completed	No

🗓️ Thursday <09 May>

	Name	Priorities 🧐	Progress 😊	Problems 😐
1	@Chloe_Duan	Update user management based on the client's feedback	Completed	No
2	@yijun liu	Overview UI design	Completed	No
3	@Junhao KONG	Redesign sidebar UI	Completed	No
4	@Yuncong Ji	Implement big file-handling API	Completed	No

5	@Zhihao Liang	Optimise RabbitMQ module	Completed	No
6	@Lingyi Kong	Update script API	Completed	No

🗓️ Tuesday <14 May>

	Name	Priorities 🧐	Progress 😊	Problems 😐
1	@Chloe_Duan	Connect the APIs for frontend and backend	Completed	No
2	@yijun liu	Connect overview part API	Completed	No
3	@Junhao KONG	Connect hide script API	Completed	No
4	@Yuncong Ji	Update database structure	Completed	No
5	@Zhihao Liang	Implement notification module	Completed	No
6	@Lingyi Kong	Implement hide script API	Completed	No

🗓️ Monday <20 May>

	Name	Priorities 🧐	Progress 😊	Problems 😐
1	@Chloe_Duan	Prepare final presentation and live demo	Completed	No
2	@yijun liu	project background of the final presentation	Completed	No
3	@Junhao KONG	user case of the final presentation	Completed	No
4	@Yuncong Ji	further improvement of the final presentation	Completed	No
5	@Zhihao Liang	product architecture of the final presentation	Completed	No
6	@Lingyi Kong	project management of the final presentation	Completed	No

🗓️ Thurday <23 May>

	Name	Priorities 🧐	Progress 😊	Problems 😐
1	@Chloe_Duan	1. Re-structure meetings notes	Completed	No

		2. Log of client communication		
2	@yijun liu	1. Action item and due dates 2. Modify overview	Completed	No
3	@Junhao KONG	1. Document spring 3 review 2. Document sprint 3 retrospective	Completed	No
4	@Yuncong Ji	1. Deployment explaination	Completed	No
5	@Zhihao Liang	1. Client meeting, stand-up meeting documentation	Completed	No
6	@Lingyi Kong	1. Trello workflow documentation	Completed	No

Sprint 4 Planning Meeting

🗓️ Date

Jun 1, 2024

👥 Participants

- @Chloe_Duan
- @Zhihao Liang
- @Yuncong Ji
- @Lingyi Kong
- @Junhao KONG

🥅 Goals

- Set up tasks for handover session

🗣️ Discussion topics

Time	Item	Notes

	Final presentation video	• Motivation 30s @yijun liu -1 min
		  ◦ Python installation
		• Product introduction/overview(features) @Lingyi Kong Our solution
		• Product demo (think about what feature to include) @Junhao KONG
		  ◦ **Script execution**
		  ◦ **Execution history**
		  ◦ Notification system **2min30s**
		• Client feedback 30s @Chloe_Duan
		• Editing @Yuncong Ji

	Code section	• Readme Product screenshot
		• Readme Refactor @Chloe_Duan
		  ◦ Descript + screenshot
		  ◦ Put How to run sectio right below demo link

✅ Action items

- [x] Update Data schema @Yuncong Ji
- [x] Contact supervisor about submitted final presentation slices(whether is the presented one) @Chloe_Duan
- [x] Database migration to local @Zhihao Liang

Decisions

Assign video task to group member and each of us is responsible for video material collection and script writing

- Final presentation section: Motivation, Product introduction/overview, Product demo, Client feedback
- Set estimated timing of video for 3.30 mins
- Using this page Product demo script to organise team member’s script

Product demo script

Motivation

Welcome to Py Runner, a web interface developed for the Radiation Oncology Department at Austin Health. Austin Health team is dedicated to advancing cancer treatment using sophisticated imaging devices like CT and MRI, combined with linear accelerators to deliver high-energy radiation precisely where it's needed.

In their mission to treat what's hidden inside the body with something invisible, they rely on a suite of Python scripts. These scripts perform essential tasks such as copying and editing files, ensuring our treatments are as efficient and accurate as possible.

However, managing these scripts poses challenges, as team members may lack Python knowledge or remote access, complicating version monitoring and central execution. Py Runner addresses these issues by allowing seamless script execution from any device without requiring Python expertise or installation.

Solution

Our solution! An All-in-One web application, PyRunner, is to simplify Python script execution. Users can effortlessly upload input files, such as medical images, and run Python scripts with just one or two clicks, regardless of their Python expertise.

Our platform offers four key functionalities:

- Script Execution - Enables non-Python users to run scripts anywhere.
- Script Management - Allows easy administration and management of shared scripts.
- Execution Management - Provides a seamless way to check execution history and outputs.
- User Management - Supports flexible team management for better collaboration.

Our web app aims to streamline script execution and management, making it accessible to all users.

Feature demo

Sign-in / user =>

overview

Most - feature - script managment and execution

Admin user add, hide, modify, delete

Execution → notification system => 2

Execution history

Client feedback

Sprints

Sprint 1

- Sprint 1 Planning

Sprint 2

- Sprint 2 planning
- Sprint 2 Review
- Sprint 2 Retrospective

Sprint 3

- Sprint 3 Planning
- Sprint 3 Review
- Sprint 3 Retrospective

Sprint 4

- Sprint 4 Planning

Sprint 1

Sprint 1 Planning

Sprint 1 Planning

Key content

- Sprint 2 plan
- Sprint 3 plan
- Technologies proposed to use
- Risk assessment
- Infrastructure to deploy

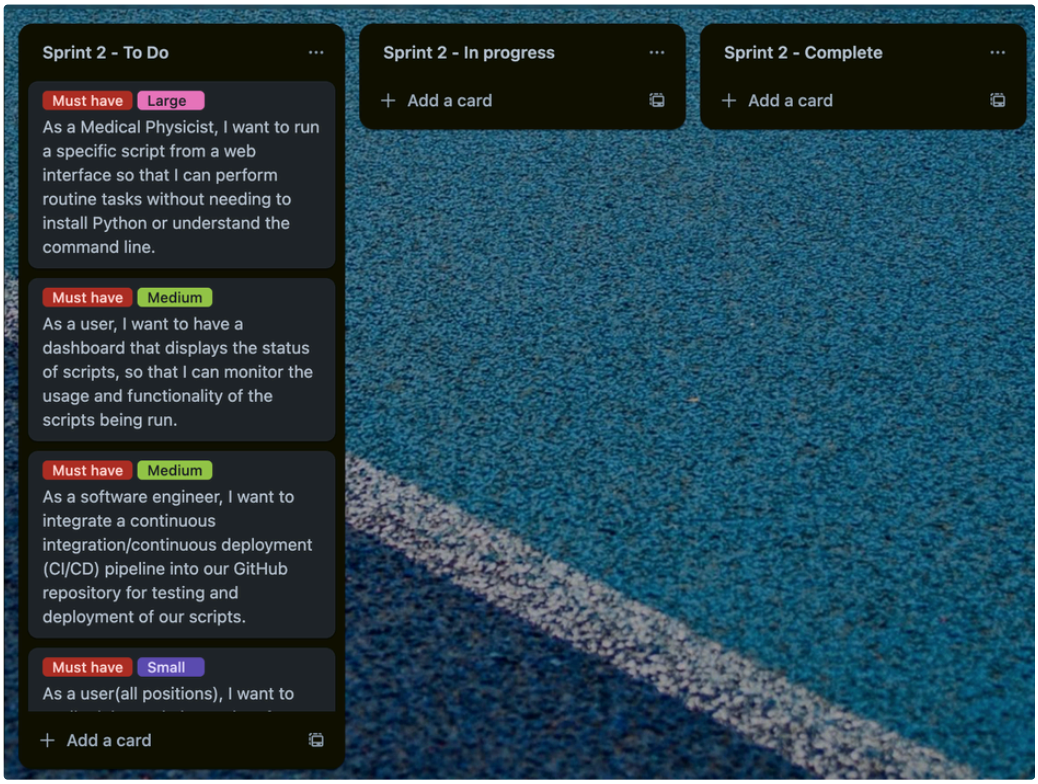
Link of our Trello board: FL-agile trello planning

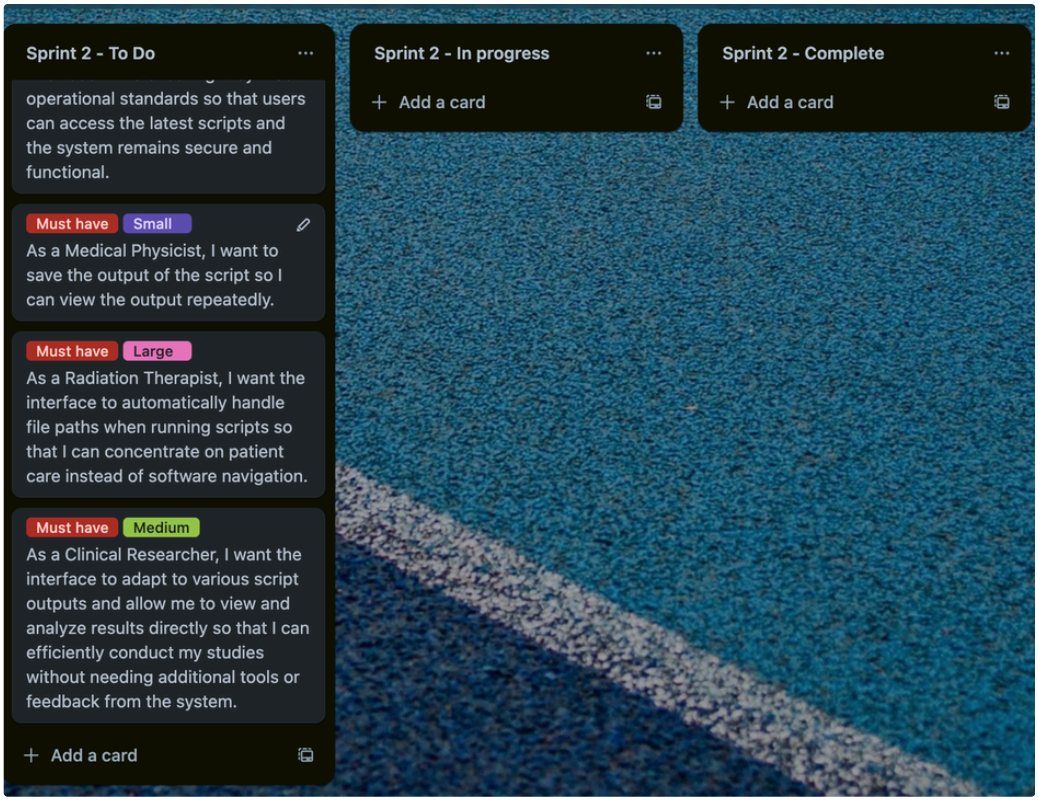
Sprint 2 Plan:

In Sprint 2, our focus is on implementing functionalities critical to the client's needs, primarily addressing user stories labeled as "Must have." with emphasizing epic of script execution and onitoring, script management and intergrate. Additionally, we'll discuss the foundational logic design for the wedsite, particularly the logic behind implementing file path changes for scripts, which is essential for our solution's architecture.

Trello Agile Sprint Board - FL

Development Requirements:





Technologies Used:

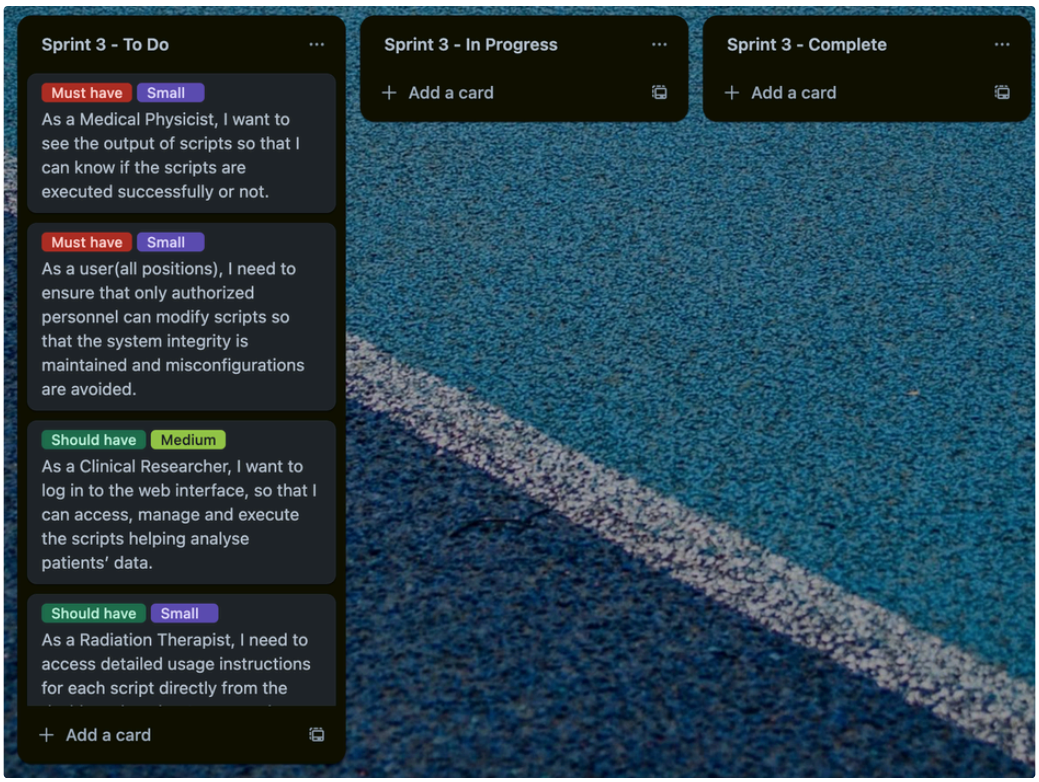
- Frontend: HTML, CSS, JavaScript (React framework)
- Backend: Flask (Python)
- Database: Mysql (for storing user information and script execution records)
- Deployment platform：Client's Healthcare Platform

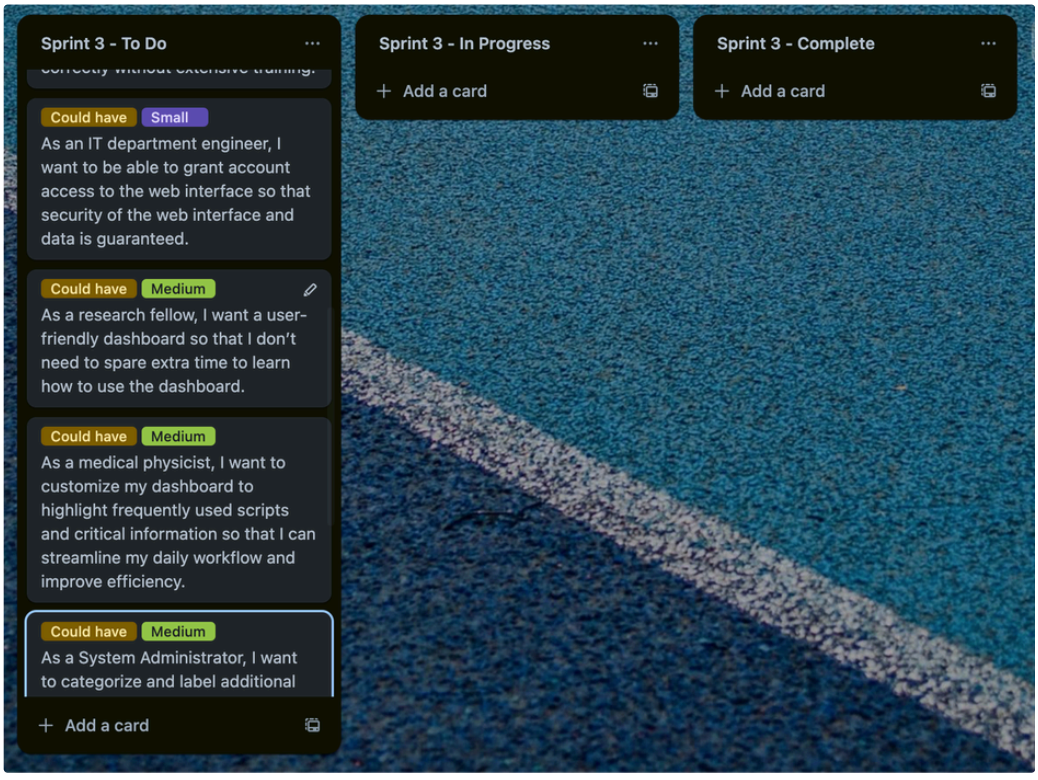
Sprint 3 Plan:

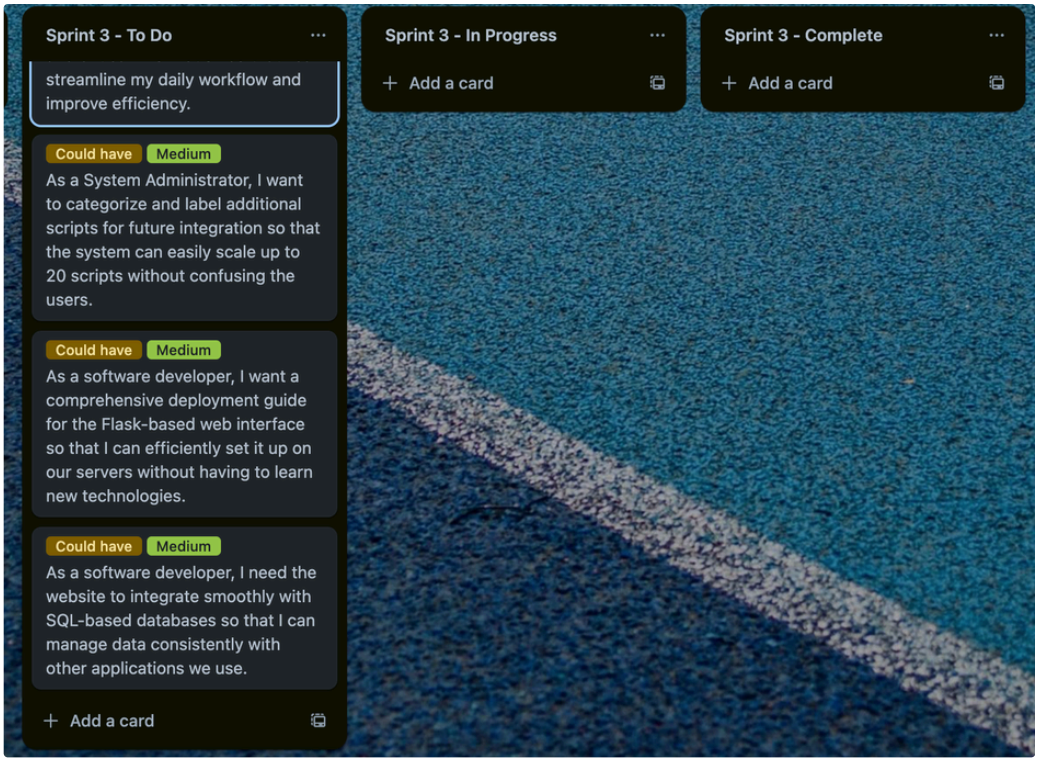
In Sprint 3, the primary focus will be on addressing the "Should Have" and "Could Have" tasks, specifically targeting improvements in the user experience and design of the dashboard. This sprint aims to enhance the overall usability and functionality of the dashboard interface, making it more intuitive and efficient for users. The team will concentrate on refining features, optimizing layout, and incorporating feedback to ensure the dashboard meets user expectations and facilitates smoother interaction with the system. By prioritizing these enhancements, we seek to elevate the user journey and provide a more engaging and effective platform for managing and monitoring tasks within the radiation oncology department's script execution and management system.

Development Requirements:

Trello Agile Sprint Board - FL


Sprint 3 - To Do
Must have Small
As a Medical Physicist, I want to see the output of scripts so that I can know if the scripts are executed successfully or not.
Must have Small
As a user(all positions), I need to ensure that only authorized personnel can modify scripts so that the system integrity is maintained and misconfigurations are avoided.
Should have Medium
As a Clinical Researcher, I want to log in to the web interface, so that I can access, manage and execute the scripts helping analyse patients' data.
Should have Small
As a Radiation Therapist, I need to access detailed usage instructions for each script directly from the
Add a card
Sprint 3 - In Progress
Add a card
Sprint 3 - Complete
Add a card


Sprint 3 - To Do
Could have Small
As an IT department engineer, I want to be able to grant account access to the web interface so that security of the web interface and data is guaranteed.
Could have Medium
As a research fellow, I want a user-friendly dashboard so that I don't need to spare extra time to learn how to use the dashboard.
Could have Medium
As a medical physicist, I want to customize my dashboard to highlight frequently used scripts and critical information so that I can streamline my daily workflow and improve efficiency.
Could have Medium
As a System Administrator, I want to categorize and label additional
Add a card
Sprint 3 - In Progress
Add a card
Sprint 3 - Complete
Add a card



Technologies Used:

- Frontend: HTML, CSS, JavaScript (React framework)
- Backend: Flask (Python)
- Database: Mysql (for storing user information and script execution records)
- Deployment platform：Client's Healthcare Platform

Risk Assessment

Given our situation, all team members are part-time developers, balancing internships or other coursework. This presents a risk: coordinating daily stand-up meetings may be challenging due to varying schedules, potentially impacting communication efficiency.

Furthermore, significant progress is anticipated mainly during weekends, which could delay client feedback and subsequently, our responses. This staggered working pattern necessitates a flexible approach to project management and communication.

Moreover, some team members are not well-versed in the Flask framework, as specified by the client, indicating a learning curve that could affect development speed and quality. This inexperience is a risk that we need to mitigate through additional training or allocating more time for those tasks.

Overall, we must adapt our development process to accommodate part-time availability and varying expertise levels, ensuring continuous progress and effective communication despite these challenges.

Infrastructure to deploy

1. **Hardware Resources**
    - **Servers**: The customer opts to use their own internal servers as cloud servers. These servers have sufficient computing power and memory to meet the needs of the medical platform.

- **Storage Space**: Data storage is managed using the customer's own servers. This may involve configuring RAID arrays or something to enhance data reliability and access speed, along with backup mechanisms to prevent data loss.

2. **Software Environment**
   - **Operating System**: A popular Windows Server version is chosen as the operating system, offering a graphical interface and broad support.
   - **Python Environment**: Python and the Flask framework are installed for web application development.
3. **Network Resources**
   - **SSL Certificate**: An SSL certificate is configured to enable HTTPS encrypted communication for the website. This is particularly important for protecting the transmission of medical information.
4. **Services and Tools**
   - **CI/CD Tools**: GitHub Actions are used for continuous integration and continuous deployment, automating the testing and deployment processes to improve development efficiency and code quality.
5. **Security Measures**
   - **Access Control**: Identity verification and access control mechanisms are implemented to ensure that only authorized users and medical professionals can access the backend management interface and sensitive data.

Sprint 2

In Sprint 2, our focus is on implementing functionalities critical to the client's needs, primarily addressing user stories labeled as "Must have." with emphasizing epic of script execution and onitoring, script management and intergrate. Additionally, we'll discuss the foundational logic design for the wedsite, particularly the logic behind implementing file path changes for scripts, which is essential for our solution's architecture.

Sprint 2 Planning

Sprint 2 Review

Sprint 2 Retrospective

Sprint 2 Planning

Driver	@Chloe_Duan @Lingyi Kong
Approver	@yijun liu
Contributors	@Chloe_Duan @Yuncong Ji @Zhihao Liang @Junhao KONG @Lingyi Kong @yijun liu
Informed	@Wei Wang
Objective	Make plans on dev tools to be used, features to be implemented, problems that might happen, milestones to be achieved.
Due date	May 2
Key outcomes	A web interface includes login / dashboard/ script execution/ profile pages and interacts with data in database
Status	NOT STARTED / IN PROGRESS / COMPLETE

🤔 Problem Statement

As we embark on this sprint, our team faces the multifaceted challenge of integrating new technologies into our project under a tight deadline. The dual pressures of academic commitments and internships have the potential to lead to uneven workloads, with some team members at risk of becoming overburdened.

Our project structure divides the development team into two focused groups: one on frontend tasks and the other on backend tasks. While this specialization enhances the flexibility of the project, it also presents the risk of integration difficulties between the two subsystems. Synchronizing the frontend and backend components will be crucial and may require additional coordination and robust integration testing.

Additionally, the diversity of operating systems used by our team members—ranging from Windows to Mac—introduces potential compatibility issues. It is imperative that we devise a development environment and workflow that are platform-agnostic, ensuring seamless collaboration and uniformity in the deployment process across varying configurations.

The upcoming sprint will concentrate on not only advancing the project but also establishing practices that mitigate these risks. We will focus on fostering communication across the frontend and backend teams, allocating time for thorough integration testing, and standardizing our development practices to accommodate the variety of operating systems in use. By proactively addressing these concerns, we aim to maintain a balanced workload among team members and ensure the successful progression of the project.

🛠️ Prototype Design

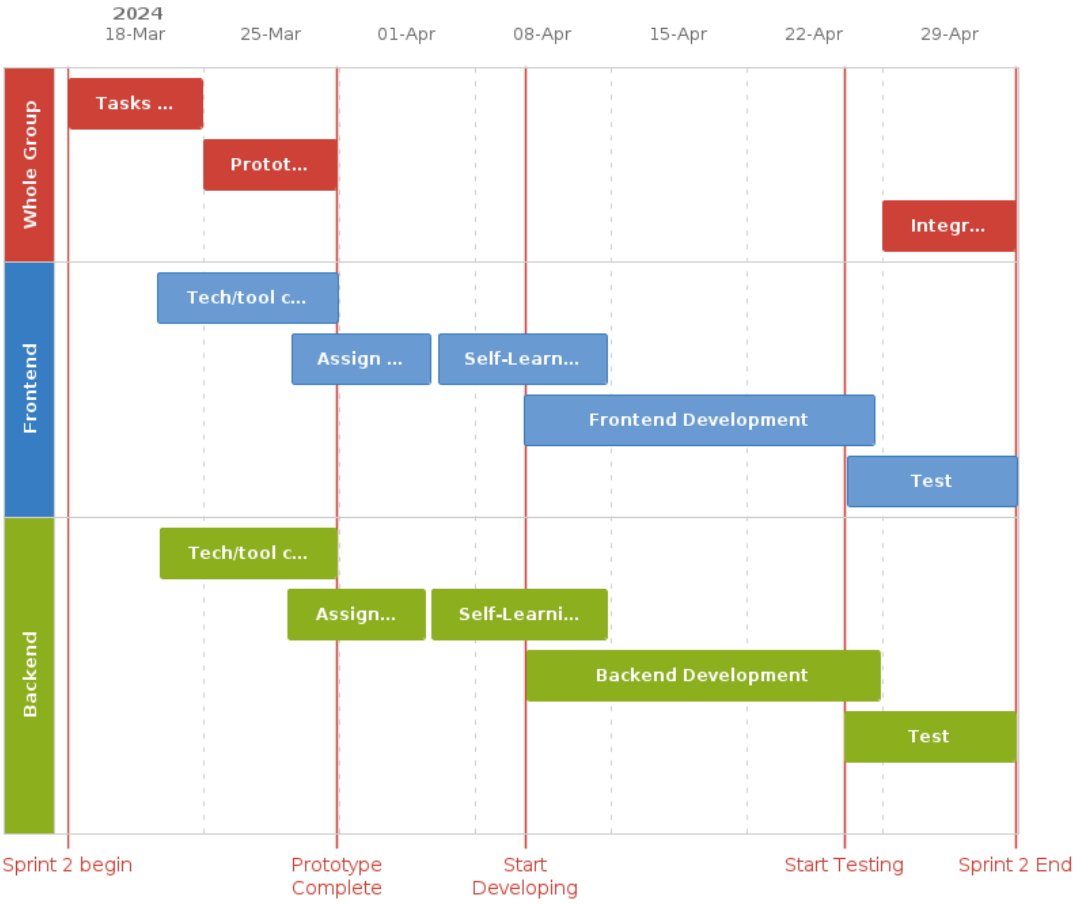
FL -Koala

🎯 Scope

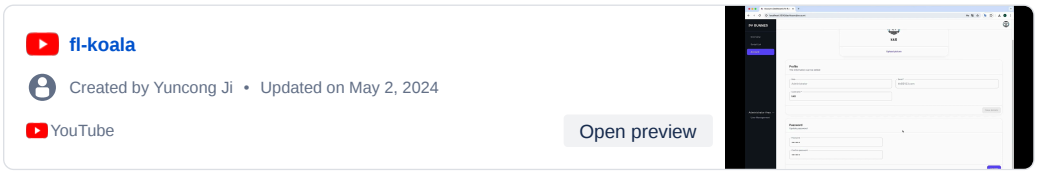
	As	I want to	So that	Task - Dev level
1	Radiation therapist	Run a specific script from a web interface easily	I can perform analysis tasks using python from any location without local installations or configurations	• Develop a backend service to execute scripts and handle session persistence for long-running tasks. • Integrate front-end interfaces with the backend to enable script execution from a web interface. • Develop a web interface for uploading, modifying, and executing scripts. • Develop automated tests for your scripts.
2	Radiation therapist	Support multiple files or folders uploading	I can analysis multiple parents information simultaneously	• Implement a file upload service on the backend that supports specific file formats and sizes. • Develop file upload functionality on the frontend. • Create a file management system to organize and store uploaded files securely.
3	Radiation therapist	Obtain the output of scripts	I can get wanted one or lots of processed patients information	• Develop a backend function to capture and store script outputs in a structured format. • Implement real-time output streaming to the frontend during script execution, including sections on the web page for success messages, error notifications, and detailed results.
4	IT department developer	Website to integrate smoothly with SQL-based databases	I can manage data consistently with other applications we use.	• Design and set up a SQL database schema that supports application needs. • Ensure data consistency and integrity across different application components.
5	Radiation therapist	Have a user-friendly script execution dashboard	I can easily find and run the script I want	• Design a dashboard that aggregates script execution statistics and history. • Implement interactive elements in the dashboard for better user engagement.
6	Radiation therapist	Access detailed usage instructions for each script directly	I can use scripts efficiently and correctly without extensive training	• Develop a context-sensitive help system that displays usage instructions based on the user's actions and roles. • Integrate instructions within the application to guide users through complex script setups.
7	Radiation therapist	Be able to save frequently used scripts	I can easily and conveniently execute the scripts I need	• Implement a feature to mark scripts as favorites and quickly access them from the dashboard. • Develop backend support for user preferences that remembers and prioritizes

				frequently used scripts.
8	Radian Researchers	version control all my scripts	I access history scripts and manage all scripts easily	• Provide a user interface for version history, diff views, and rollback options.
9	Department administrator	Have two groups of users, administrator and normal user	Coworkers with proficient coding knowledge can upload, edit and delete needed scripts and the other group can focus on script execution	• Develop a comprehensive user management system that supports two roles (Administrator & User) with different permissions. • Implement role-based access control within the application to ensure secure operations. • Create an administrator role in the web interface to oversee user management, including roles assignment, access permissions, create and delete user. • Implement features for account management, enabling users to update profiles and settings.
10	Department administrator	Let different users access the corresponding system based on their role	They can access different features based on their permission.	• Create dynamic user interfaces that adjust based on the user's role and permissions. • Ensure that API endpoints enforce role-based access controls strictly.
11	Department administrator	Have a role of administrator for user account management	The efficient management of users can be handles within teams.	• Implement administrative functions to manage user roles, permissions, and access controls.

🗓️ Timeline



Project Demonstration



https://www.youtube.com/watch?v=pmYtdWO1FxU

Milestones and deadlines

Milestone	Owner	Deadline	Status
Finalise dev tool selection.	@Chloe_Duan	Apr 1, 2024	COMPLETE
Assign tasks for frontend and backend.	@Chloe_Duan @Zhihao Liang	Apr 1, 2024	COMPLETE
Finish prototype development.	@Lingyi Kong @yijun liu	Apr 1, 2024	COMPLETE
Frontend and backend development.	Dev team	Apr 28, 2024	COMPLETE

Complete frontend testing	@Chloe_Duan @Junhao KONG @yijun liu	Apr 28, 2024	COMPLETE
Complete backend testing	@Zhihao Liang @Yuncong Ji @Lingyi Kong	May 1, 2024	COMPLETE
Complete Integration testing.	Dev team	May 2, 2024	COMPLETE
Finalise all sprint documentations.	Dev team	May 2, 2024	COMPLETE
Finalize all sprint tasks	Dev team	May 2, 2024	COMPLETE

🔗 Reference materials

Sprint 2 Review

Duration	Mar 23, 2024 to May 2, 2024
Goal	Demonstrate and review the work completed during the Sprint.
Team	FL-Koala
Participants	@Chloe_Duan @Zhihao Liang @yijun liu @Yuncong Ji @Junhao KONG @Lingyi Kong
Informed	@Wei Wang
Agenda	Sprint Goal Sprint Deliverables Incomplete works Challenges Performance Review • Burndown Chart • Performance Analysis • Actions to do Demonstration Next Steps Plan Feedback and Discussion

🏁 Sprint Goal

In this Sprint, our objective is to build and implement the essential user stories that establish the core features of our online script execution platform. By adopting a frontend-backend separation approach, we aim to finalize the login, dashboard, user profiles, script execution, and script overview functionalities. Our focus is to ensure smooth data access and manipulation from the database, as well as reliable online script execution capabilities for our users.

🎯 Sprint Deliverables

- We developed a feature that allows administrators to upload scripts in zip file formats, ensuring a seamless setup and preparation for script execution without manual file management.
- We implemented a user-friendly web interface that enables users to execute specific scripts directly from the web, bypassing the need for local Python installations or command-line interactions.
- We enabled real-time monitoring of script execution, allowing users to view/download outputs instantly, which helps in quick decision-making and troubleshooting.
- A secure login system was established to manage user authentication and script operations securely, providing role-based access control to ensure that only authorized users can perform sensitive operations.

- We provided administrative tools for managing user roles and permissions, streamlining the process of user management within teams and ensuring system integrity.
- The interface was enhanced to adapt to various output types, facilitating users in their studies by allowing them to analyze script outputs directly without needing additional software tools.
- A dashboard was developed to give users real-time data on script status, enhancing transparency and operational oversight.
- We integrated a CI/CD pipeline into our GitHub repository to automate the testing and deployment processes, ensuring that all script updates are smoothly transitioned into production.
- User stories included developments for easy script uploading, deleting, and saving functionalities, enhancing the overall user experience and maintaining the security and integrity of the platform.

🗒️ Incomplete Work

While we made substantial progress this sprint, certain enhancements remain incomplete but are crucial for improving user interaction with our platform. Our next steps include refining the web interface to enhance clarity and user-friendliness. Specifically, we aim to implement a dynamic script information overview on the homepage, which will display newly updated scripts, the most popular scripts, and those recently modified. This feature will allow users to quickly access and assess scripts based on current relevance and community engagement.

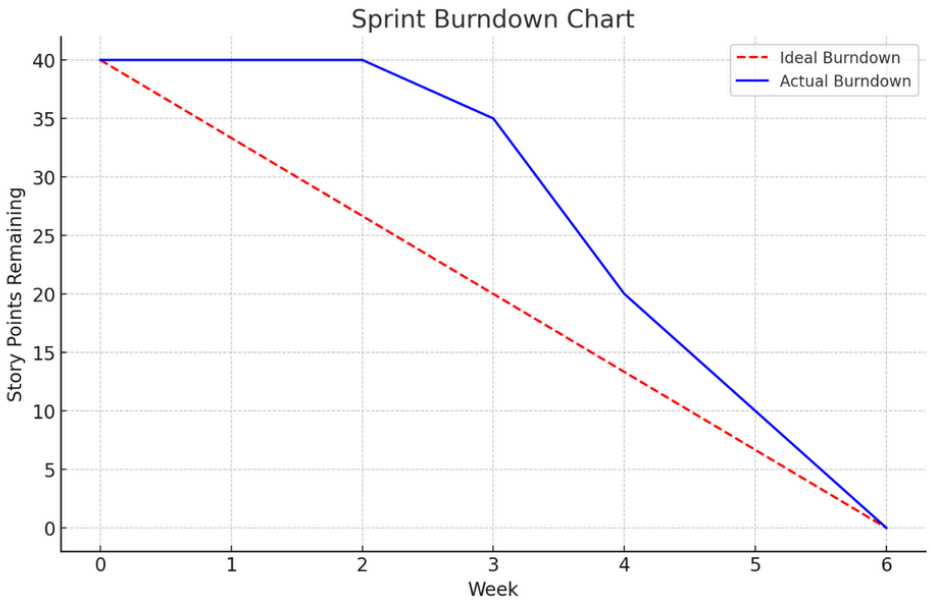
Additionally, we plan to integrate functionality to allow users to favorite or unfavorite scripts. This will streamline the user experience, enabling one-click operations to customize their interface and interact more efficiently with the scripts they need most often. These features are designed to not only personalize the user experience but also to enhance the accessibility and efficiency of the platform, making it more intuitive for all users.

⛰️ Challenges

Throughout this Sprint, we faced several challenges that tested our adaptability and resilience. Balancing our full-time studies with professional internships and side jobs, our team managed the complex task load with varying degrees of success. Initially, the frontend group had to scale a steep learning curve to familiarize themselves with the new technologies that underpin our development. Additionally, diverse operating systems and development tools across team members' laptops presented integration hurdles. Varied progress in frontend and backend tasks also led to synchronization difficulties, at times resulting in backend development outpacing frontend efforts.

⏮️ Performance Review

- **Burndown Chart**



- **Performance Analysis**

The team's performance depicted in the Burndown Chart highlights a delayed start, with no story points completed in the first two weeks, indicating a potential struggle with either task initiation or encountering early impediments. However, the performance surged in week 4 and continued steadily in weeks 5 and 6, culminating in the completion of all story points by the end of the sprint. This pattern suggests a period of adjustment followed by accelerated productivity, with the team demonstrating an ability to catch up and fulfill sprint commitments despite initial delays.

- **Action to do**

1. Conduct a thorough review of sprint planning and task breakdown to ensure tasks are actionable from the outset.
2. Address any obstacles or complexities early on, possibly through more focused problem-solving sessions or clearer prioritisation.
3. Reinforce the importance of maintaining a consistent work pace throughout the sprint to avoid a heavy workload towards the end.
4. Recognise and analyse successful strategies and practices that led to the eventual completion of goals to replicate these in future sprints.
5. Continue to celebrate the team's successes and learn from each sprint's unique challenges and outcomes.

💻 Demonstrations

Sprint 2 demo

⏭️ Next Steps Plan

For the upcoming Sprint, we plan to continue our progress by completing the remaining 'Should-have' and 'Could-have' user stories, thus enriching our web service's functionality. Furthermore, we will refine the user interface to enhance its aesthetics and user experience. By the conclusion of the next Sprint, we intend to integrate all components cohesively and execute a comprehensive integration test. This will ensure that our application operates seamlessly across different user interactions.

😀 Feedback and Proposals

Our clients did not attend the sprint review meeting as they went on vacation and will not come back to work until `May 13, 2024`. Therefore, we did not have chance to show our work to them and get feedbacks from them at the end of this sprint. We plan to schedule a meeting with the clients to show our deliverables in sprint 2 and ask for feedbacks on how to adjust our web interface.

However, we did take a meeting inner our group for sprint review, and each team members expressed their opinions to make it more smooth and opportune to complete the upcoming sprint 3.

- There was a shared agreement on the challenges faced during the integration of frontend and backend systems. The team discussed the need for better synchronisation mechanisms, as highlighted in the sprint planning documentation, which could prevent potential delays and ensure smoother integration in future sprints.
- Feedback from the frontend team emphasized the steep learning curve for new technologies, which was also noted in the sprint retrospective. Participants appreciated the gradual improvement in handling these technologies towards the end of the sprint. The need for additional training sessions and resources was suggested to accelerate the learning process and reduce the initial lag in productivity.
- As documented in the sprint planning, the diverse operating systems used by the team members posed significant compatibility challenges. The discussion underscored the necessity of establishing a protocol for regular compatibility checks, which was a proposed action item in the retrospective to prevent such issues in the future.
- The enhancements made to the user interface and script management functionalities were well-received, as seen in the sprint review documentation. However, feedback also pointed towards the need for continuous improvement in the UI to cater to the evolving needs of the users, particularly in simplifying script management tasks.

Feedback

In the client meeting on May 16, we got the feedback from the client. The client agrees with our current design of selecting the input path and finds our interface design to be very reasonable. Some of the features we delivered exceeded client expectations. In addition, the client provided modified suggestions for the code implementation of our script, believing that we could write for loop function for the script. We will try to implement this in future deliveries.

Proposals

Code Implementation

- Review the current script to identify areas where a for loop can enhance functionality and efficiency.
- Refactor the code to include the for loop, ensuring it meets the client’s requirements.
- Test the modified script thoroughly to ensure it works as expected without introducing new issues.

Interface Design

- Maintain the current design for selecting the input path as the client is satisfied with it.

Sprint 2 Retrospective

📋 Overview

Date	May 1, 2024
Team	FL-Koala
Participants	@Chloe_Duan @Zhihao Liang @Lingyi Kong @yijun liu @Yuncong Ji @Junhao KONG

💭 Retrospective

Start doing	Stop doing	Keep doing
• Enhance cross-team communication between frontend and backend groups.	• Cease the practice of postponing task initiation.	• Holding regular meetings.
• Allocate adequate time for task completion.	• Avoid overlooking or dismissing suggestions.	• Offer mutual support among team members.
• All members actively contribute to the project's success.	• Eliminate casual attitudes toward development.	• Maintain in-group communication.
• Identify problems early and address issues proactively.	• Refrain from disconnecting from team activities.	• Follow the scrum development process

✅ Action items

Actions	Owner
• Cross-team weekly Sync-up: Initiate a weekly sync-up meeting between the frontend and backend teams to ensure alignment and early identification of integration issues. Set up the initial schedule and invite all necessary team members.	@Lingyi Kong
• Task Start Timeline: Develop a task start timeline to prevent the late initiation of tasks. Integrate this into the project management tool by the next sprint planning session, ensuring tasks have clear start dates.	@Junhao KONG @yijun liu
• Idea Review Sessions: Schedule regular idea review sessions to ensure all team member suggestions are heard and evaluated. This can be included as part of the regular meetings. Incorporate these sessions into the existing meeting structure.	@Chloe_Duan
• Development Standards Workshop: Conduct a workshop to reinforce development standards and the importance of maintaining a professional	@Zhihao Liang

approach to work. Organise the workshop and prepare relevant materials.	
• Issue Tracking System: Implement or improve the use of an issue tracking system to ensure problems are reported, tracked, and resolved in a timely manner. Select an appropriate tool and set up a process for the team to follow.	@Yuncong Ji
• Compatibility Check Protocol: Establish a protocol for regular compatibility checks across different operating systems used by the team to catch and resolve issues early. Create a checklist and schedule for these checks, ensuring that every piece of code is tested on both Windows and Mac environments before being merged.	@Zhihao Liang

Sprint 3

In previous sprint, we successfully implemented all 'must have' user stories but identified several logical bugs impacting performance and user experience. Sprint 3 will focus on resolving these issues to stabilize and refine our application. Additionally, we will implement selected 'non-must have' user stories from our product backlog to enhance functionality. Key improvements will include detailed script information and the capability for batch script processing, aiming to improve user efficiency and interface intuitiveness. This sprint is critical for transitioning our system from functional to polished, ready for broader deployment and use.

Sprint 3 Planning

Sprint 3 Review

Sprint 3 Retrospective

Sprint 3 Planning

Driver	@Chloe_Duan @Lingyi Kong
Approver	Austin hospital client
Contributors	@Chloe_Duan @yijun liu @Zhihao Liang @Yuncong Ji @Lingyi Kong @Junhao KONG
Informed	@Wei Wang
Objective	1. Improve the bugs and logic loopholes appeared last sprint 2. Complete new proposed feedback and requirement from client 3. Implement some non-highly prioritized tasks
Due date	May 25 2024
Key outcomes	Finalised released product
Status	FINISHED

🤔 Problem Statement

In Sprint 2, we successfully implemented all 'must have' user stories but identified several logical bugs impacting performance and user experience. Sprint 3 will focus on resolving these issues to stabilize and refine our application. Additionally, we will implement selected 'non-must have' user stories from our product backlog to enhance functionality. Key improvements will include detailed script information and the capability for batch script processing, aiming to improve user efficiency and interface intuitiveness. This sprint is critical for transitioning our system from functional to polished, ready for broader deployment and use.

🎯 Scope

ID	As	I want to	So that	Task
5	Radiation therapist	Have a user-friendly script execution dashboard	I can easily find and run the script I want	1. Display the creator’s full name in the script table
12	Medical physicist	Have a dashboard that displays the status of scripts	I can have orginased space to run script and see which script to be run.	1. Implement each user corresponding status of scripts 2. Design script status showing table 3. API connection between frontend and backend
13	Radiation researcher	Categorize and label additional scripts for future integration	The system can easily scale up to 20 scripts without confusing the users	1. Enable user to define different label contents 2. Enable user to add custormised label to different scripts
14	Medical physicist	Customize my dashboard to highlight frequently used scripts and critical information	I can streamline my daily workflow and improve efficiency.	1. Modular UI component of overview page

				2. Enable user to drag or choose their component
15	IT Departmentdeveloper	Have a comprehensive deployment guide for the Flask-based web interface	I can efficiently set it up on our servers without having to learn new technologies.	Whrite necessary deploy document
16	Clinical Oncologist	Schedule scripts to run at specific times or intervals	Routine tasks can be automated, reducing manual effort and increasing consistency.	1. Enable set timer to recurrly execute scripts
17	Medical physicist	Manage user information flexibly	I can streamline user management and improve efficiency.	1. Login as username with 6 digit string or number 2. reset all user's password in one click 3. Have a new field called fullname 4. Each new user has initial password: 123456
18	Radiation researcher	Be notified when long script executions are done.	I can know when my Python script execution is complete.	1. Develop a system to monitor long-running script execution status.Ensure accurate tracking of script progress and completion. 2. Create a system to trigger notifications upon script completion. Enable notifications via multiple channels (e.g., email, in-app). 3. Modify the interface to display notification settings and alerts. Allow users to configure their notification preferences easily.
19	Radiation researcher	Have a dashboard that displays the status of scripts, access to previous execution outputs, and view my execution history.	I can have an organized space to run scripts, review my past operations, and know what operations I have done.	1. Create a dashboard to display the status of scripts. Ensure the dashboard is user-friendly and allows users to see which scripts need to be run. 2. Implement a feature to access and review previous execution outputs. Ensure outputs are easily accessible even if the original downloads are lost or users are on different devices. 3. Create a new section to display the user's execution history. Ensure both frontend and backend are updated to support this feature.

Milestones and deadlines

Milestone	Owner	Deadline	Status
Assign tasks for frontend and backend.	@Chloe_Duan @Zhihao Liang	May 3, 2024	COMPLETE

Frontend and backend development.	Dev team	May 20, 2024	COMPLETE
Complete frontend testing	@Chloe_Duan @Junhao KONG @yijun liu	May 23, 2024	COMPLETE
Complete backend testing	@Zhihao Liang @Yuncong Ji @Lingyi Kong	May 23, 2024	COMPLETE
Complete Integration testing.	Dev team	May 23, 2024	COMPLETE
Finalise all sprint documentations.	Dev team	May 23, 2024	COMPLETE
Finalize all sprint tasks	Dev team	May 23, 2024	COMPLETE

Sprint 3 Review

Duration	May 3, 2024 to May 23, 2024
Goal	Demonstrate and review the work completed during the Sprint.
Team	FL-Koala
Participants	@Chloe_Duan @Zhihao Liang @yijun liu @Yuncong Ji @Junhao KONG @Lingyi Kong
Informed	@Wei Wang
Agenda	Sprint Goal Sprint Deliverables Challenges Performance Review • Burndown Chart • Performance Analysis • Actions to do Next Steps Plan Feedback and Discussion

🏁 Sprint Goal

The primary goal of Sprint 3 was to resolve logical bugs identified in Sprint 2 and implement selected 'non-must have' user stories to enhance the application's functionality. This includes developing a dashboard for script status, providing access to previous execution outputs, managing execution history, notifying users of long script executions, and ensuring smooth integration with SQL-based databases. Key improvements targeted user efficiency, interface intuitiveness, and overall system stability, transitioning our application from functional to polished, ready for broader deployment and use.

🎯 Sprint Deliverables

1. **Bug Fixes:**
    - **Batch Adding Favorite Scripts:**
        - Enabled the functionality to add favorite scripts in batch mode.
        - Modified the process to avoid returning related script data after adding a favorite script.
    - **Backend Enhancements:**
        - Completed search and filter functionalities, shifting them from frontend to backend.
        - Addressed the issue of no records for the last script running time.
        - Implemented solutions to handle long-running scripts and logout issues.
    - **Frontend Enhancements:**
        - Developed a dedicated page to display the status of running scripts.

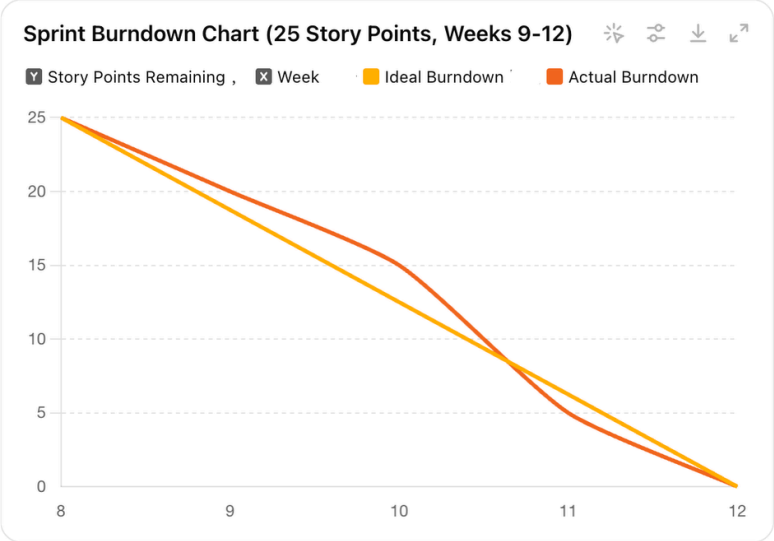
        - Implemented error handling for 404 errors.
    - **Backend & Frontend Enhancements:**
        - Allowed administrators to hide scripts by setting their status to active/inactive.
        - Improved script fetching speed.
        - Ensured that deleting a user no longer removes their scripts from the system.
2. **Implemented User Stories:**
    - **ID 12: Medical Physicist's Dashboard**
        - Implemented a dashboard to display the status of scripts, organizing the space for script execution.
        - Designed a table to show the status of each user's scripts.
        - Established API connections between the frontend and backend for script status updates.
    - **ID 13: Script Categorization and Labeling**
        - Enabled users to define and add custom labels to different scripts, facilitating future scalability.
    - **ID 14: Customizable Dashboard**
        - Developed a modular UI component for the overview page.
        - Allowed users to customize their dashboard by dragging or choosing components, highlighting frequently used scripts and critical information.
    - **ID 15: Deployment Guide**
        - Created a comprehensive deployment guide for the Flask-based web interface, aiding efficient setup on servers.
    - **ID 16: Script Scheduling**
        - Enabled users to schedule scripts to run at specific times or intervals, automating routine tasks.
    - **ID 17: Flexible User Management**
        - Enabled login using a 6-digit string or number as username.
        - Implemented a feature to reset all users' passwords with a single click.
        - Added a new field called 'fullname' for users.
        - Set an initial password '123456' for each new user.
    - **ID 17: Flexible User Management**
        - Enabled login using a 6-digit string or number as username.
        - Implemented a feature to reset all users' passwords with a single click.
        - Added a new field called 'fullname' for users.
        - Set an initial password '123456' for each new user.
    - **ID 18: Notification System**
        - Monitor long-running script status.
        - Track script progress and completion.Implement Backend Monitoring.
        - Develop Notification Mechanism.
        - Display notification settings and alerts.
    - **ID 19: Status and History management**
        - Developed a user-friendly dashboard which sisplays script status.
        - Have access to previous execution outputs
        - Add new section for execution history.
3. **Other Improvements:**
    - Displayed the creator’s full name in the script table to enhance user-friendliness.

⛰️ Challenges

Implementing notifications for long-running script executions presents significant challenges. The primary difficulty lies in the architectural modifications required to accurately track and monitor script execution times. This involves developing a reliable system that can continuously monitor the status of scripts and trigger notifications upon their completion. The backend needs substantial updates to handle real-time tracking and data processing without impacting performance. Additionally, the frontend must be capable of displaying these notifications in a user-friendly manner, ensuring that users are promptly informed when their scripts finish execution. Balancing these updates while maintaining the integrity and performance of the existing system is a complex task. Ensuring seamless integration of this feature without introducing new issues or delays is crucial for the overall success of the project.

⏮️ Performance Review

- **Burndown Chart**



- **Performance Analysis**

The sprint burndown chart for weeks 9 to 12 with a total of 25 story points shows a steady reduction in story points, closely following the ideal burndown line. This indicates effective sprint planning and execution by the team. The actual burndown line aligns well with the ideal burndown, reflecting the team's consistent progress in completing user stories and resolving issues. The smooth decline in story points suggests that the team managed their workload efficiently and tackled tasks systematically. There were no significant spikes or plateaus, indicating that the team did not encounter major obstacles or delays during this period.

- **Action to do**

1. **Conduct a Retrospective:** Hold a sprint retrospective to discuss what went well, what could be improved, and any lessons learned. This will help the team identify best practices and areas for improvement.
2. **Review Task Estimations:** Evaluate the accuracy of task estimations and adjust the estimation process if needed to ensure future sprints are planned effectively.
3. **Maintain Consistent Communication:** Encourage ongoing communication within the team to promptly address any issues that arise, ensuring continued smooth progress in future sprints.
4. **Focus on Continuous Improvement:** Identify any minor issues or inefficiencies observed during the sprint and develop strategies to address them, fostering a culture of continuous improvement.

5. **Prepare for the Next Sprint:** Based on the success of this sprint, plan and prioritize tasks for the next sprint, ensuring a balanced workload and clear objectives for the team.

💻 Demonstrations

Sprint 3 demo

⏭️ Next Steps Plan

1. Schedule user testing sessions.
2. Collect and analyze user feedback.
3. Continue monitoring the application for any issues.
4. Plan for the next sprint, focusing on handover session and potential bugs.

😀 Feedback and Proposals

Feedback:

Our clients attended the sprint review meeting and reviewed our product. They expressed satisfaction with the progress and the quality of our deliverables. They provided valuable feedback, particularly on the user interface and script management functionalities, and were pleased with the enhancements made. The clients appreciated the improvements and offered suggestions for further refinement to better meet their needs.

Proposal:

1. **Prepare Handover:**
   - Develop comprehensive documentation covering all aspects of the project, including codebase, system architecture, and user guides.
   - Create a detailed project timeline and checklist to facilitate a smooth handover process, ensuring all deliverables are completed and verified before the final handover date.
2. **Incorporate Client Positive Feedback:**
   - Review the feedback received from clients during the sprint review and prioritize the suggested improvements.
   - Implement the changes and enhancements recommended by the clients to further refine the user interface and overall system functionality.
3. **Improve Bugs and Logic Loopholes:**
   - Conduct a thorough review of the system to identify and document all existing bugs and logic loopholes.
   - Implement automated testing and regular code reviews to detect and prevent future bugs and logic issues, ensuring a more stable and reliable system.

Sprint 3 Retrospective

📋 Overview

Date	May 23, 2024
Team	FL-Koala
Participants	@Chloe_Duan @Zhihao Liang @Lingyi Kong @yijun liu @Yuncong Ji @Junhao KONG

💭 Retrospective

Start doing	Stop doing	Keep doing
• **Drop Unnecessary Features with Consideration of Time:** Prioritize essential features and functionalities, dropping those that are not critical to the project's success to better manage time and resources.	• **Developing Large and Challenging Unnecessary Features:** Avoid dedicating time and resources to developing complex features that are not essential to the project's core requirements. Focus on delivering the most important functionalities first.	• **Adherence to Scrum Principles:** Continue following the scrum framework diligently to ensure structured and efficient project management.
• **Proactive Issue Resolution:** Encourage team members to identify and address issues as soon as they arise, rather than waiting for formal meetings.	• **Inactive Meetings and Team Contribution:** Cease holding unproductive meetings where team contribution is minimal. Ensure all meetings have clear agendas and active participation from all members.	• **Collaborative Problem Solving:** Maintain the culture of mutual support and collaboration, ensuring all team members contribute to solving problems.
• **Allocate Buffer Time:** Build buffer time into the sprint schedule to accommodate unexpected challenges without compromising deadlines.	• **Overreliance on Online Meetings:** Avoid relying solely on online meetings for communication. When possible, encourage face-to-face interactions to enhance collaboration and understanding.	• **Client Engagement:** Keep the clients involved and informed, scheduling regular check-ins to gather feedback and adjust as necessary.

✅ Action items

Actions	Owner
• **Enhance Team Synchronization:**	@Lingyi Kong

◦ Schedule regular cross-team sync meetings to discuss integration points and address any discrepancies between frontend and backend work. ◦ Use collaborative tools for better communication and task tracking across teams.	
• **Proactive Issue Management:** ◦ Implement a system for early detection and resolution of issues, possibly through regular code reviews and automated testing. ◦ Encourage a proactive approach where team members feel empowered to address problems as they identify them..	@Junhao KONG @yijun liu
• **Proactive Issue Management:** ◦ Implement a system for early detection and resolution of issues, possibly through regular code reviews and automated testing. ◦ Encourage a proactive approach where team members feel empowered to address problems as they identify them.	@Chloe_Duan
• **Buffer Time Allocation:** ◦ Integrate buffer time into the sprint planning to handle unexpected challenges without affecting the overall timeline. ◦ Review and adjust buffer time allocation based on the complexity of tasks and previous sprint experiences.	@Zhihao Liang
• **Early Task Initiation:** ◦ Ensure that tasks are started promptly at the beginning of the sprint to avoid last-minute rushes and ensure steady progress. ◦ Monitor task initiation closely and take corrective actions if delays are observed.	@Yuncong Ji
• **Client Feedback Loop:** ◦ Schedule follow-up meetings with clients post-sprint review to gather their feedback on the latest deliverables. ◦ Use client feedback to make iterative improvements and ensure the product meets their expectations and needs.	@Zhihao Liang

Sprint 4

In this sprint, we focus on finalizing our product according to the client's feedback from the final presentation and handing over the finalized product to the client. We discussed with the client and decided to use a zip format to deliver the product, including the product code, detailed deployment guidance, and final presentation slides. Our team will support the client with further deployment after the handover procedure.

Sprint 4 Planning

Sprint 4 Product Finalisation

Sprint 4 Planning

Driver	@Chloe_Duan @Lingyi Kong
Approver	Austin hospital client
Contributors	@Chloe_Duan @yijun liu @Zhihao Liang @Yuncong Ji @Lingyi Kong @Junhao KONG
Informed	@Wei Wang
Objective	`Improve the bugs and logic loopholes that appeared in the last sprint` `Handover preparsion`
Due date	June 7 2024
Key outcomes	Finalised released product
Status	TO DO

🤔 Problem Statement

Sprint 4 Planning

In Sprint 4, our main goals are to resolve remaining bugs and prepare for handover. Key objectives include:

1. **Bug Fixes:**
   - Address all outstanding issues.
   - Conduct thorough testing.
2. **Handover Preparation:**
   - Create comprehensive documentation.
   - Conduct knowledge transfer sessions.
   - Finalize and document the deployment process.
3. **User Experience:**
   - Refine the UI based on feedback.
   - Ensure all features are functional and user-friendly.
4. **Performance Optimization:**
   - Optimize performance and scalability.
   - Monitor and adjust for load performance.
5. **Final Polishing:**
   - Review and clean up the codebase.
   - Ensure adherence to coding standards.

This sprint is critical for delivering a polished product ready for deployment.

🎯 Scope

Improve for implemented user stories:

Backend:	1. Large file upload 2. Improve notification module	
Frontend:	1. Large file upload	
Backend & frontend:	1. Support docker deployment	

Sprint 4 backblog:

Epic	As	I want to	So that	Priority	Size	Task
Handover	IT department developer	Website to integrate smoothly with SQL-based databases	I can manage data consistently with other applications we use.	Must have	Small	The IT department of the client company has requested an easy-to-use SQL-based database, a common requirement in the technology sector.
	IT department developer	Website to integrate smoothly with SQL-based databases	I can manage data consistently with other applications we use.	Must have	Small	The IT department of the client company has requested an easy-to-use SQL-based database, a common requirement in the technology sector.

Sprint 4 Product Finalisation

Key information

In this sprint, we focus on finalizing our product according to the client's feedback from the final presentation and handing over the finalized product to the client. We discussed with the client and decided to use a zip format to deliver the product, including the product code, detailed deployment guidance, and final presentation slides. Our team will support the client with further deployment after the handover procedure.

🎯 Sprint Deliverables

1. **Schema Preparation**: Begin by exporting the schema from the cloud database. This involves generating SQL scripts that define the structure of the database, including tables, views, indexes, and stored procedures. Tools and scripts should be selected or developed to automate this process to ensure accuracy and efficiency.
2. **Seed SQL File Preparation**: Create a seed SQL file that populates the local database with a set of initial data necessary for application operation. This file is crucial for environments like development or testing where representative data is needed for functionality testing.
3. **Local Environment Setup**: Set up the local database server with the necessary software and configurations. This includes installing database management software, configuring storage, and setting access permissions, ensuring the local environment mirrors the cloud setup as closely as possible for testing purposes.

😀 Feedback and Proposals

Feedback

We have sent the final release to the client and received positive feedback. The client appreciated the product and the detailed deployment guidance provided.

Proposal

To ensure a smooth transition and deployment, we propose the following support after the handover:

1. **Ongoing Support**:
   - Our team will be available to assist with any deployment issues or questions the client may have.
   - We will provide support for configuring the local environment and resolving any setup challenges.
2. **Documentation Updates**:
   - Continuously update the deployment documentation based on any feedback received during the deployment process to ensure clarity and completeness.
3. **Regular Check-ins**:
   - Schedule regular check-ins with the client to address any concerns and provide necessary updates or support.

Project Architecture

In this section, project architecture is illustrated including

- Technologies
- Database schema
- Deployment
- Project UML

Technologies

For the development of our website, we have opted for a decoupled architecture, using TypeScript for the front-end and Python for the back-end. This approach allows us to leverage the strengths of both languages, ensuring a robust and scalable solution.

TypeScript has been chosen as the front-end language due to its ability to overcome several limitations of JavaScript. As a superset of JavaScript, TypeScript introduces strong typing, enhanced code structuring, and object-oriented features, which significantly reduce the likelihood of runtime errors and improve maintainability. This is particularly advantageous for projects regardless of their size, and it has gained substantial popularity within the Australian IT industry for its reliability and the robust tooling ecosystem.

On the back-end, we have implemented the Flask framework to meet the specific needs of our client. The client's familiarity with Flask, combined with its straightforward and lightweight nature, makes it an ideal choice for our project. Flask's ability to integrate seamlessly with other Python libraries and tools allows for quick development cycles and easy maintenance. This framework supports our needs for a flexible and efficient server-side solution that can easily adapt to changing requirements.

This technology stack not only facilitates a smooth development process but also ensures that both the client and our team can manage and scale the project effectively. The use of these modern, well-supported technologies positions us to handle the dynamic demands of web applications efficiently.

Frontend Technologies

Programming Language:

TypeScript

- **Role**: A superset of JavaScript providing optional static typing.
- **Benefits**: Facilitates building more robust software, while reducing the likelihood of common JavaScript errors at runtime.

Library:

1. **React and React-DOM**
    a. **Role**: Library for building user interfaces, particularly single-page applications where data updates frequently.
    b. **Benefits**: Facilitates the creation of reusable UI components, manages state effectively across components, and optimizes rendering performance.
2. **Next.js**
    a. **Role**: A React framework that provides server-side rendering, static site generation, and optimized bundling. It is used to build scalable and performant React applications.
    b. **Benefits**: Simplifies routing, API routes creation, and overall project structure. It also enhances performance with automatic code splitting and image optimization.

Styling and UI Frameworks

1. **Material-UI**
    - **Role**: A popular React UI framework with a comprehensive set of pre-designed components that follow Material Design principles.
    - **Benefits**: Accelerates development by providing ready-to-use components like buttons, cards, dialogs, etc., which are customizable and responsive.

Additional Libraries and Utilities

- **Axios**: Promise-based HTTP client for making requests to back-end services.
- **DayJS**: Lightweight date-time library, used for parsing, validating, and manipulating dates and times.

Backend Technologies

1. **Flask**
   - **Role**: Flask is a lightweight WSGI web application framework in Python, designed to be easy to use and extend, making it ideal for building simple web applications quickly and scaling up to complex applications.
   - **Benefits**: Since Flask is not prescriptive about database or ORM, it allows developers to choose their tools and libraries, which provides flexibility in building the application architecture.
2. **Flask-JWT-Extended**
   - **Role**: This extension provides JWT (JSON Web Tokens) support for Flask applications, facilitating secure and scalable authentication mechanisms.
   - **Benefits**: Offers features like token refreshing, token revocation, and additional security settings that are crucial for modern API security.
3. **SQLAlchemy**
   - **Role**: SQLAlchemy is an ORM (Object Relational Mapping) tool that allows Python applications to communicate with databases using high-level abstractions.
   - **Benefits**: Manages database schemas and performs database operations in a Pythonic way, reducing the need for raw SQL unless necessary and boosting productivity.

Additional Libraries and Utilities

- **Flask-Cors**: A Flask extension for handling Cross-Origin Resource Sharing (CORS), making it safe to enable cross-origin requests in the Flask app when necessary, especially for APIs consumed by different domains.

Development and Operations

- **Docker (Dockerfile and docker-compose.yml)**: These files suggest that Docker is used to containerize the application, ensuring it runs consistently across different environments.
- **Blinker**: Provides support for Signal handling in Python, which can be used to handle events in the application, like modifications in the database or certain actions triggering other actions.

Cloud Database: Supabase

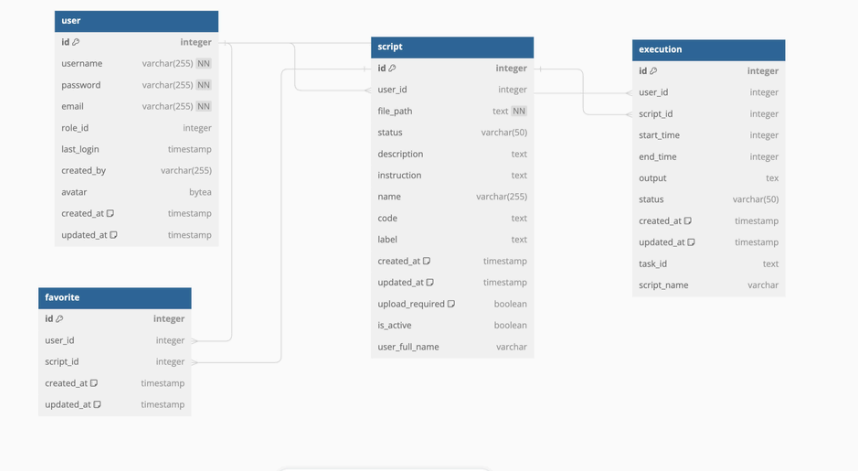
Supabase is an open-source Firebase alternative that provides a suite of tools including a PostgreSQL database, authentication, instant APIs, real-time subscriptions, and storage. It's designed to simplify the backend development process, offering many Firebase-like features but with the flexibility of a traditional SQL database.

Local Database: PostgreSQL

PostgreSQL is a powerful, open-source object-relational database system. It uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. PostgreSQL comes with features such as advanced optimization, strong reliability, and robust access-control mechanisms, which make it highly extensible and compliant with the SQL standard. PostgreSQL is highly customizable with stored procedures, triggers, and support for a wide variety of indexing methods, which makes it an ideal choice for local database management and operations.

Database schema

Database relationtionship diagram is shown below.



Database url:

```
postgresql+psycopg2://postgres.bgniwlymuiboukuqbaha:LFofHUle88xRS6sv@aws-0-ap-southeast-2.pooler.supabase.com:5432/postgres
```

```
// Use DBML to define your database structure
// Docs: https://dbml.dbdiagram.io/docs

Table user {
  id integer [primary key]
  username varchar(255) [unique, not null]
  password varchar(255) [not null]
  email varchar(255) [unique, not null]
  role_id integer
  last_login timestamp
  created_by varchar(255)
  avatar bytea
  created_at timestamp [default: 'CURRENT_TIMESTAMP']
  updated_at timestamp [default: 'CURRENT_TIMESTAMP']
}

Table script {
  id integer [primary key]
  user_id integer
  file_path text [not null]
  status varchar(50)
  description text
  instruction text
  name varchar(255)
  code text
  label text
  created_at timestamp [default: 'CURRENT_TIMESTAMP']
  updated_at timestamp [default: 'CURRENT_TIMESTAMP']
  upload_required boolean [default: false]
  is_active boolean
  user_full_name varchar
}

Table execution {
  id integer [primary key]
  user_id integer
  script_id integer
  start_time integer
```

```
  end_time integer
  output tex
  status varchar(50)
  created_at timestamp [default: 'CURRENT_TIMESTAMP']
  updated_at timestamp [default: 'CURRENT_TIMESTAMP']
  task_id text
  script_name varchar
}

Table favorite {
  id integer [primary key]
  user_id integer
  script_id integer
  created_at timestamp [default: 'CURRENT_TIMESTAMP']
  updated_at timestamp [default: 'CURRENT_TIMESTAMP']
}

Ref: script.user_id > user.id // many-to-one
Ref: execution.script_id > script.id // many-to-one
Ref: execution.user_id > user.id // many-to-one
Ref: favorite.user_id > user.id // many-to-one
Ref: favorite.script_id > script.id // many-to-one
```

Deployment

Background

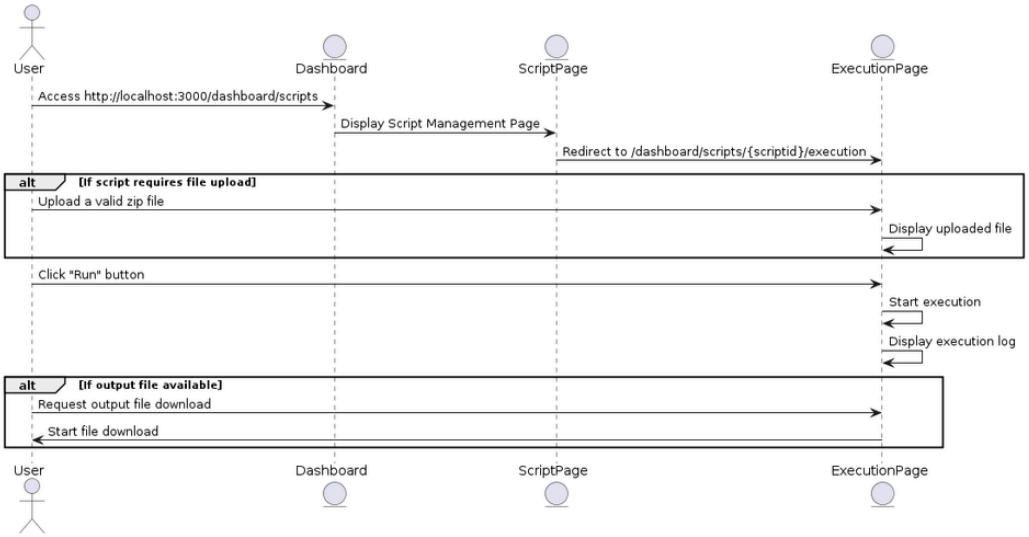
The current project involves developing a sophisticated application intended to be deployed on the client’s internal servers. As of now, the project has not been publicly deployed. This document outlines the primary reasons for this decision, as per the client's requirements and strategic objectives.

Reasons for Non-Deployment

1. Client’s Strategic Decision:
   The primary reason the project has not been deployed publicly is due to the client's strategic decision to run the application on their own servers. The client has specific operational needs and security protocols that necessitate an internal deployment.
2. Security Concerns:
   The nature of the application involves handling sensitive data, and the client has stringent security policies. Deploying the application on their own servers allows the client to maintain full control over the data, ensuring that it remains within their secure environment. This approach minimizes the risk of data breaches and aligns with their compliance requirements.
3. Integration with Existing Systems:
   The application needs to integrate seamlessly with the client's existing systems and databases. Hosting the application internally facilitates smoother integration, allowing for direct communication with other services and databases that are part of the client’s IT ecosystem. This ensures better performance and reliability.

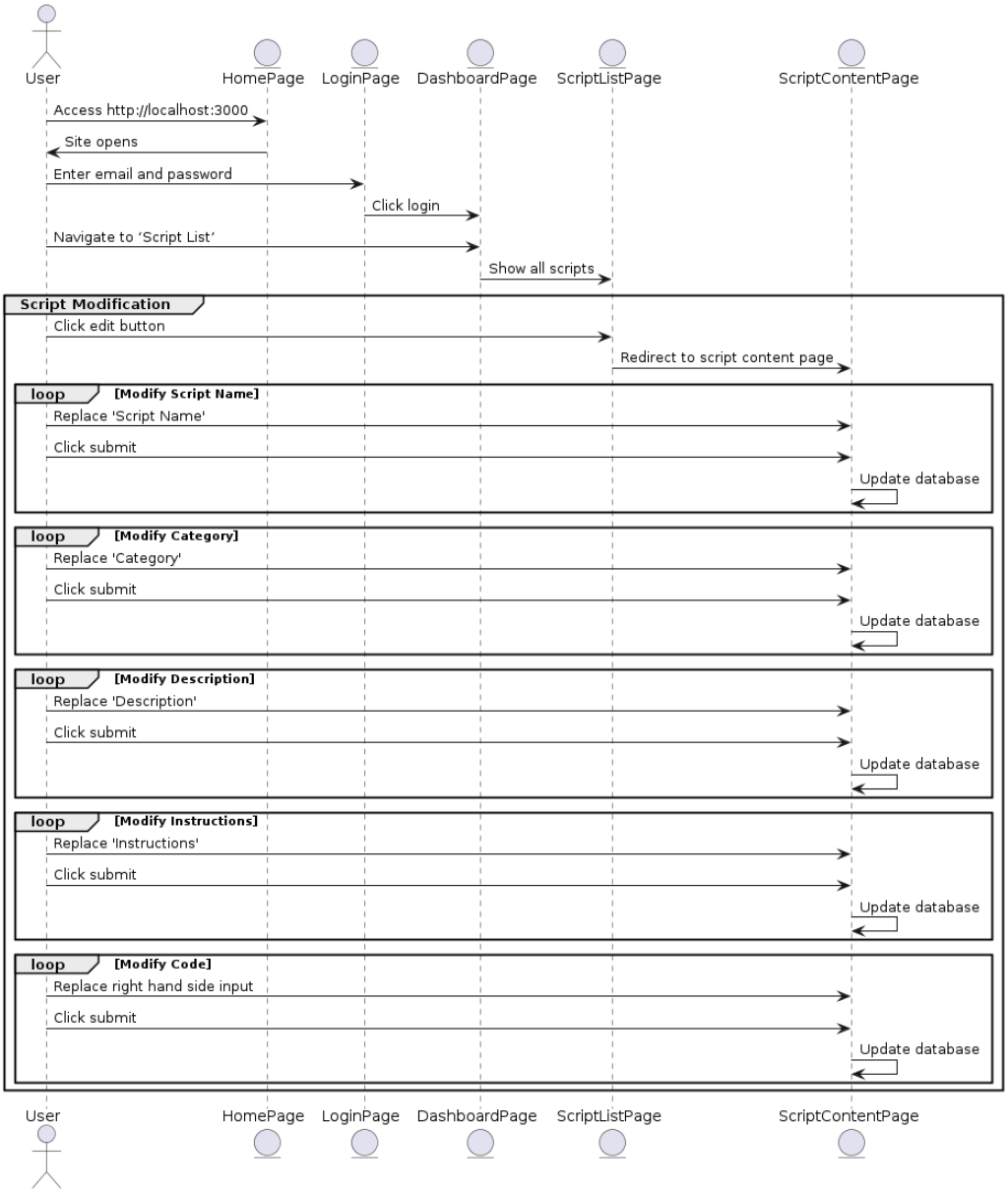
Project UML

Script Execution SD




```
@startuml
actor User
entity Dashboard
entity ScriptPage
entity ExecutionPage

User -> Dashboard: Access http://localhost:3000/dashboard/scripts
Dashboard -> ScriptPage: Display Script Management Page

ScriptPage -> ExecutionPage: Redirect to /dashboard/scripts/{scriptid}/execution

alt If script requires file upload
    User -> ExecutionPage: Upload a valid zip file
    ExecutionPage -> ExecutionPage: Display uploaded file
end

User -> ExecutionPage: Click "Run" button
ExecutionPage -> ExecutionPage: Start execution
ExecutionPage -> ExecutionPage: Display execution log

alt If output file available
    User -> ExecutionPage: Request output file download
    ExecutionPage -> User: Start file download
end

@enduml

```

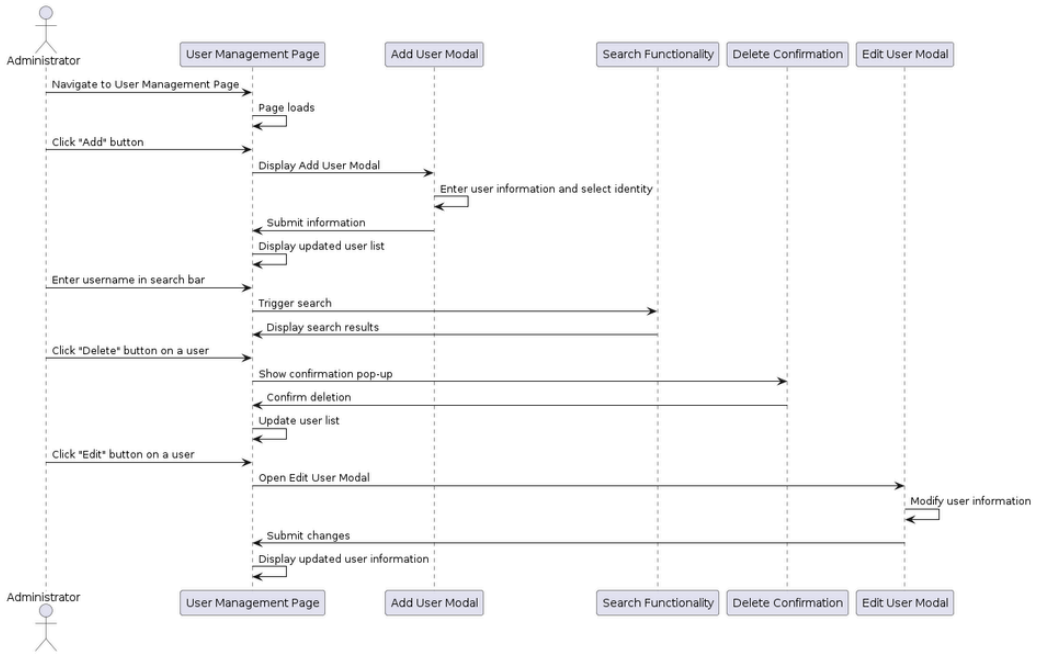
Edit Script SD




```
@startuml
actor User
entity HomePage
entity LoginPage
entity DashboardPage
entity ScriptListPage
entity ScriptContentPage

User -> HomePage: Access http://localhost:3000
HomePage -> User: Site opens

User -> LoginPage: Enter email and password
LoginPage -> DashboardPage: Click login
```

```
14
15  User -> DashboardPage: Navigate to ‘Script List’
16  DashboardPage -> ScriptListPage: Show all scripts
17
18  group Script Modification
19      User -> ScriptListPage: Click edit button
20      ScriptListPage -> ScriptContentPage: Redirect to script content page
21
22      loop Modify Script Name
23          User -> ScriptContentPage: Replace 'Script Name'
24          User -> ScriptContentPage: Click submit
25          ScriptContentPage -> ScriptContentPage: Update database
26      end
27
28      loop Modify Category
29          User -> ScriptContentPage: Replace 'Category'
30          User -> ScriptContentPage: Click submit
31          ScriptContentPage -> ScriptContentPage: Update database
32      end
33
34      loop Modify Description
35          User -> ScriptContentPage: Replace 'Description'
36          User -> ScriptContentPage: Click submit
37          ScriptContentPage -> ScriptContentPage: Update database
38      end
39
40      loop Modify Instructions
41          User -> ScriptContentPage: Replace 'Instructions'
42          User -> ScriptContentPage: Click submit
43          ScriptContentPage -> ScriptContentPage: Update database
44      end
45
46      loop Modify Code
47          User -> ScriptContentPage: Replace right hand side input
48          User -> ScriptContentPage: Click submit
49          ScriptContentPage -> ScriptContentPage: Update database
50      end
51  end
52
53  @enduml
54
```

User management SD




```
@startuml
actor Administrator as admin
participant "User Management Page" as UMP
participant "Add User Modal" as AUM
participant "Search Functionality" as SF
participant "Delete Confirmation" as DC
participant "Edit User Modal" as EUM

admin -> UMP : Navigate to User Management Page
UMP -> UMP : Page loads

admin -> UMP : Click "Add" button
UMP -> AUM : Display Add User Modal
AUM -> AUM : Enter user information and select identity
AUM -> UMP : Submit information
UMP -> UMP : Display updated user list

admin -> UMP : Enter username in search bar
UMP -> SF : Trigger search
SF -> UMP : Display search results

admin -> UMP : Click "Delete" button on a user
UMP -> DC : Show confirmation pop-up
DC -> UMP : Confirm deletion
UMP -> UMP : Update user list

admin -> UMP : Click "Edit" button on a user
UMP -> EUM : Open Edit User Modal
EUM -> EUM : Modify user information
EUM -> UMP : Submit changes
```

```
UMP -> UMP : Display updated user information

@enduml

```

Quality Assurance

- Web Security
- Ethical Considerations
- Test
- Code Review

Web Security

Security Background：

This project aims to develop an internal website for a medical company, designed to facilitate the execution of scripts. Given the sensitive nature of patient data involved, stringent security measures are crucial to ensure data protection and compliance with regulatory standards. This document outlines the security measures implemented in the project to safeguard patient data and enhance overall security.

Security Measures Record：

Security Measure	Description	Implementation Date	Responsible Person	Review Date	Notes
Password Encryption	Use the encryption technology and methods to encrypt user passwords.	2024-04-19	@Zhihao Liang	2024-04-26	Regularly check encryption strength
Environment Variable for Database URL	Explains in detail how to use environment variables to store database URLs and other sensitive configurations to protect patient data security	2024-04-19	@Zhihao Liang	2024-04-26	Review the security of environment variables
Library Updates	Details the policy and process for regularly updating the libraries used in the project, including the tools used and the frequency of updates.	2024-04-19	@Lingyi Kong	2024-04-26	Focus on security update releases
JWT Authentication	Introduces the use of JWT for user authentication, including the processes for generating and verifying tokens.	2024-04-19	@Yuncong Ji	2024-04-26	Monitor vulnerabilities in authentication mechanisms
Frontend Access Control	Implement measures to ensure that non-admin users cannot access admin sections of the site, such as blocking direct URL access	2024-05-10	@Chloe_Duan	2024-05-23	Regularly test access control mechanisms

Details:

1. **Encrypting Passwords**: By encrypting passwords, you ensure that even if data is intercepted or accessed by unauthorised individuals, the passwords remain protected. This helps prevent unauthorised access and mitigates potential damage.
2. **Using Environment Variables for Database URLs**: Storing sensitive information such as database URLs in environment variables, rather than in your codebase, reduces the risk of exposing them in source control or to unauthorized access through other means. This is a standard practice for maintaining the security of database connections.
3. **Using Latest Libraries**: Keeping your libraries up-to-date is crucial because many updates include patches for security vulnerabilities that have been discovered since the last version. Regular updates help protect your systems from known attacks that exploit these vulnerabilities.
4. **Using JWT for Authentication**: JSON Web Tokens (JWT) are a compact, URL-safe means of representing claims to be transferred between two parties. Using JWTs can help in securely transmitting information between the client and server, which is especially useful for stateless authentication.
5. **Frontend Access Control ：**To prevent non-admin users from accessing admin sections of the site, security measures have been implemented to block direct URL access. For example, if a non-admin user tries to access `localhost://3000/admin`, they will be redirected or denied access, ensuring that only authorized users can enter admin areas.

Maintenance:

1. **Regular Review**: The team should review the document on security measures at least once a week to ensure all information is up-to-date.
2. **Change Management**: Any changes or additions to security practices must be updated in this document. Changes must be reviewed and approved by the security team first.
3. **Access Control**: Only team members can edit this document; all other people will have read-only access.
4. **Feedback Mechanism**: Team members are encouraged to offer improvement suggestions. Feedback will be collected via team meetings or the internal email system.

Ethical Considerations

Ethical Issues and Considered

1. **Privacy**:
   - **Concern**: User scripts may contain sensitive information, which could be inadvertently exposed or misused.
   - **Resolution**: Implement rigorous data handling protocols to ensure that all user data is encrypted and securely stored. Access to this data is restricted to authorized personnel only.
   - **Justification**: Protecting user data is a fundamental ethical obligation that also helps in building trust with our users.
2. **Transparency**:
   - **Concern**: Users need to understand how their data and scripts are being used and what the outputs imply.
   - **Resolution**: Provide clear documentation and user agreements that outline how data is handled, the purpose of data collection, and how outputs are generated.
   - **Justification**: Transparency is key to ethical user interactions and helps in ensuring that users are making informed decisions.
3. **Honesty**:
   - **Concern**: Accurate representation of what the script execution can and cannot do.
   - **Resolution**: Ensure that all user communications are honest and clear about the capabilities and limitations of our service.
   - **Justification**: Honesty prevents misinformation and promotes a trustworthy relationship with users.
4. **Inclusivity**:
   - **Concern**: Ensuring that the website is accessible to all potential users, regardless of their technical skills or disabilities.
   - **Justification**: Inclusivity enhances the usability of the website for a broader audience and is a key ethical commitment.
5. **Sustainability**:
   - **Concern**: Efficient use of server resources to avoid unnecessary computational waste and reduce operational costs.
   - **Resolution**: Optimize server performance through effective load balancing, caching, and on-demand resource allocation. Implement efficient coding practices to minimize processing time.
   - **Justification**: Efficient resource management reduces costs and improves service reliability and performance, aligning with ethical standards for responsible technology use.

Test

Sprint 2 Testing

- Test case 01
- Test case 02
- Test case 03
- Test case 04
- Test case 05
- Test case 06

Sprint 3 Testing

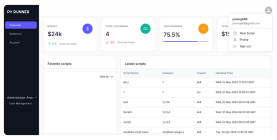
- Test case 07
- Test case 08
- Test case 09
- Test case 10
- Test case 11
- Test case 12

Sprint 2 Testing

- Test case 01
- Test case 02
- Test case 03
- Test case 04
- Test case 05
- Test case 06

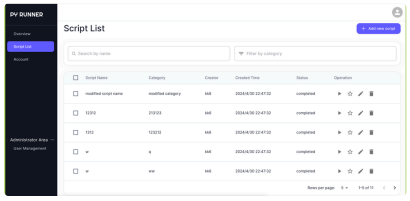
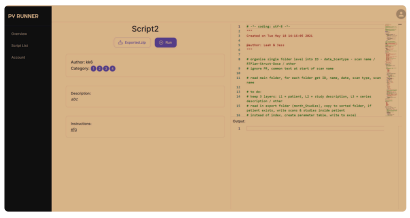
Test case 01

Test Case ID	1
Created By	@Zhihao Liang
Test Case Description	Log in to the system
Tester's Name	@Zhihao Liang
Date Tested	Apr 28, 2024
Test Case (Pass/Fail/Not Executed)	Pass

Step #	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:3000	Site should open	As Expected		Pass
2	Enter email and password email:jiutong666@gmail.com password:123456	Display email and password with placeholder	As Expected		Pass
3	Click login	Go to the dashboard page	As Expected		Pass

Test case 02

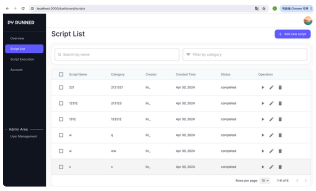
Test Case ID	2
Created By	@Zhihao Liang
Test Case Description	Execute a script from script management page
Tester's Name	@Zhihao Liang
Date Tested	Apr 30, 2024
Test Case (Pass/Fail/Not Executed)	Pass

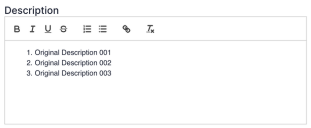
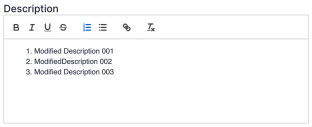
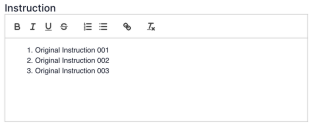
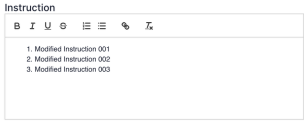
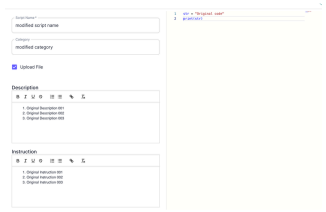
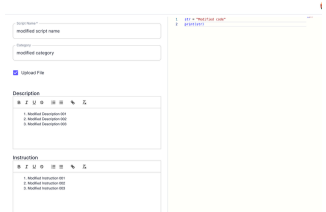
Step #	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:3000/dashboard/scripts	Script management page should open	As Expected		Pass
2	Click "Run Script" button in a row in the action column	Redirect to /dashboard/scripts/{scriptid}/execution	As Expected		Pass
3	If script requires file upload, upload a valid zip file	Upload input should accept file and display uploaded	As Expected		Pass
4	Click "Run" button	If upload required, execution starts after upload. Without upload,	As Expected		Pass

		execution starts immediately.			
5	View execution log in output area	Execution log should display progress and results	As Expected		Pass
6	If available, download output file	Download should start and file should be correct	As Expected		Pass

Test case 03

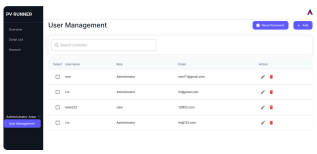
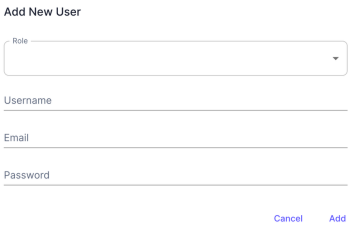
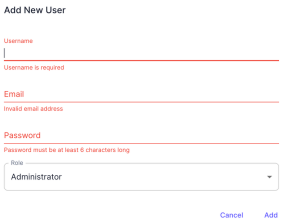
Test Case ID	3
Created By	@Lingyi Kong
Test Case Description	List available scripts and modifications on script details
Tester's Name	@Lingyi Kong
Date Tested	Apr 30, 2024
Test Case (Pass/Fail/Not Executed)	Pass

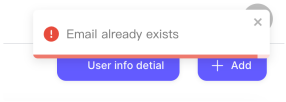
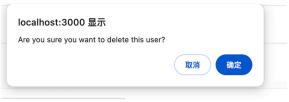
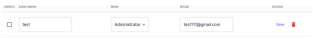
Step #	Owner	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	@Lingyi Kong	Navigate to the 'Script List' page.	Show all scripts in the 'Script List' page.	As expected		Pass
2	@Lingyi Kong	Click the edit button of the script to be modified, it will redirect to the script content page; Replace content in "Script Name" input box with desired value; Click "submit" to submit modification.	'Script name' be modified successfully and be stored in the database.	As expected	before: after: 	Pass
3	@Lingyi Kong	Click the edit button of the script to be modified, it will redirect to the script content page; Replace content in "Category" input box with desired value;	'Category' be modified successfully and be stored in the database.	As expected	before: after: 	Pass

		Click “submit” to submit modification.				
4	@Lingyi Kong	Click the edit button of the script to be modified, it will redirect to the script content page; Replace content in “Description” input box with desired value; Click “submit” to submit modification.	‘Description’ be modified successfully and be stored in the database.	As expected	before:  after: 	Pass
5	@Lingyi Kong	Click the edit button of the script to be modified, it will redirect to the script content page; Replace content in “Instruction” input box with desired value; Click “submit” to submit modification.	‘Instruction’ be modified successfully and be stored in the database.	As expected	before:  after: 	Pass
6	@Lingyi Kong	Click the edit button of the script to be modified, it will redirect to the script content page; Replace content in right hand side input box with desired value; Click “submit” to submit modification.	‘Code’ be modified successfully and be stored in the database.	As expected	before:  after: 	Pass

Test case 04

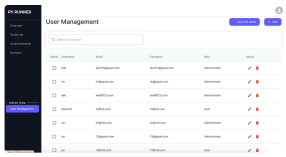
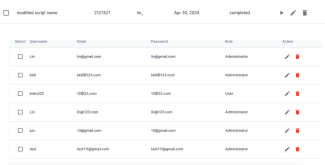
Test Case ID	4
Created By	@yijun liu
Test Case Description	Manage user information from the User Management page
Tester's Name	@yijun liu
Date Tested	Apr 30, 2024
Prerequisite	User must be logged in as administrator
Test Case (Pass/Fail/Not Executed)	Pass

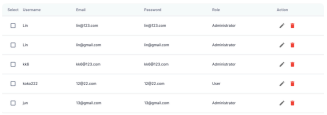
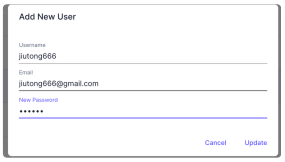
Step #	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:3000/dashboard/customers	User management page should open	As Expected		Pass
2	Click "Add" button in the upper right corner	The user information will be added after entering user information and selecting an identity	As Expected		Pass
		If the added information is empty, the information cannot be added	As Expected		
		If the added user email address already exists,	As Expected		Pass

		a message will be displayed			
3	If a specific user needs to be searched, enter the user name in the search bar to search for the specific user	Upload input should accept file and display uploaded	As Expected		Pass
4	If a user needs to be deleted，click the delete button	A pop-up window will pop up asking if you are sure to delete the user	As Expected		Pass
5	If a user's information needs to be edited or changed, click the edit button	User information can be changed directly	As Expected		Pass

Test case 05

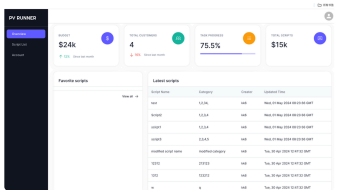
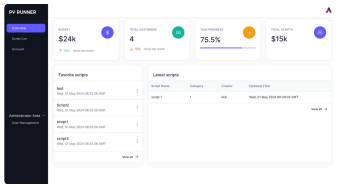
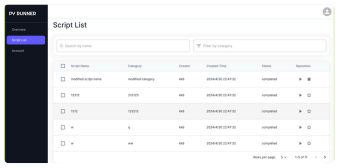
Test Case ID	5
Created By	@Yuncong Ji
Test Case Description	user account info
Tester's Name	@Yuncong Ji
Date Tested	Apr 30, 2024
Test Case (Pass/Fail/Not Executed)	Pass

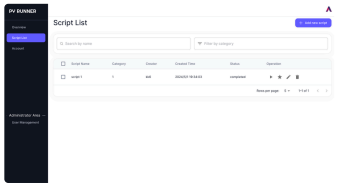
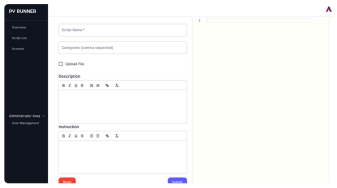
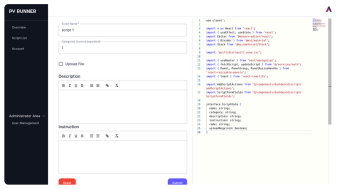
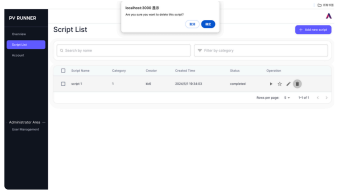
Step #	Owner	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	@Yuncong Ji	Navigate to the 'User Management' page.	Show all users in the ' User Management'' page.	As expected		Pass
2	@Yuncong Ji	Click the delete button of the user to be modified,	'user' be deleted successfully and updated in the database.	As expected	before:  after 	Pass
3	@Yuncong Ji	Click the edit button of the user to be modified Update Username,Email Password,Role	user information be modified successfully and be stored in the database.	As expected	before: after:	Pass

		Click save button to save user information				
4	@Yuncong Ji	Click “Add“ button to add new User of administrator Enter username, email, password and select user or administrator. Click “add“ to add user on page. 	“User“ be added successfully and be stored in the database.	As expected	before:  after: 	Pass
5	@Yuncong Ji	Must select button to choose user for update password click “user info detail“ to updated password click update button to save new password	‘password’ be modified successfully and be stored in the database.	As expected		Pass

Test case 06

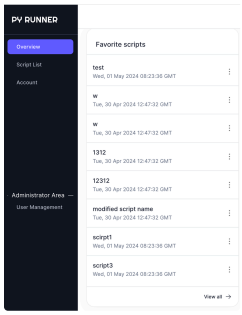
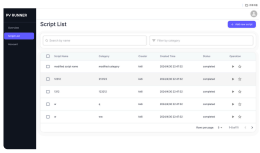
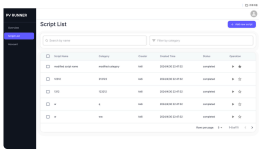
Test Case ID	6
Created By	@Junhao KONG
Test Case Description	Test different role functions (admin & user)
Tester's Name	@Junhao KONG
Date Tested	May 1, 2024
Test Case (Pass/Fail/Not Executed)	Pass

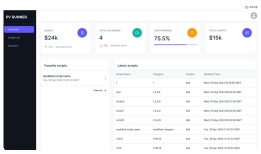
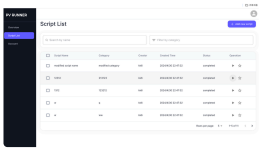
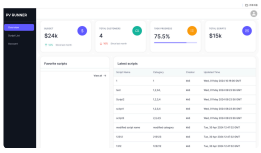
Step #	Owner	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	@Junhao KONG	Use the user account to log in. email: test111@gmail.com password:123456	Show the user page. There is no user management page.	As expected		Pass
2	@Junhao KONG	Use the admin account to log in email: kk6@123.com passowrd:123456	Show the admin page. There is a user management page.	As expected		Pass
3	@Junhao KONG	Navigate the script list page using the user account. email: test111@gmail.com password:123456	Users can not add new scripts, modify scripts, or delete scripts.	As expected		Pass

4	@Junhao KONG	Navigate the script list page using the admin account. email: kk6@123.com passowrd:123456	There is an add new script button for the admin to add new scripts. There are modify and delete buttons in the operation list.	As expected		Pass
5	@Junhao KONG	Click the Add New Scripts button using the admin account. email: kk6@123.com passowrd:123456	The add new scripts page can run successfully.	As expected		Pass
6	@Junhao KONG	Click the Modify button using the admin account. email: kk6@123.com passowrd:123456	The modify scripts page can run successfully.	As expected		Pass
7	@Junhao KONG	Click the Modify button using the admin account. email: kk6@123.com passowrd:123456	The scripts can be deleted successfully.	As expected		Pass

Test case 07

Test Case ID	7
Created By	@Junhao KONG
Test Case Description	Test favourite function script in the overview page
Tester's Name	@Junhao KONG
Date Tested	May 1, 2024
Test Case (Pass/Fail/Not Executed)	Pass

Step #	Owner	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	@Junhao KONG	Navigate to the overview page	There is a favourite scripts area in the left corner of the page.	As expected		Pass
2	@Junhao KONG	Navigate to the scripts list	There is a star button (for favourite scripts) in the operation list.	As expected		Pass
3	@Junhao KONG	Click the star button for the script called(modified script name), and navigate to the overview page.	The script called(modified script name) is shown in the favourite scripts list.	As expected		Pass

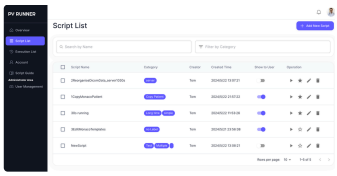
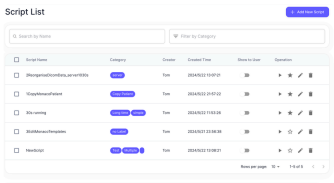
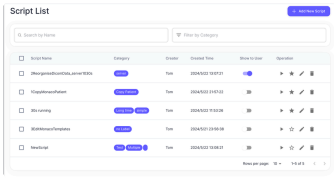
						
4	@Junhao KONG	Click the star button for the script called(modified script name) again to remove the script, and navigate to the overview page.	The script called(modified script name) is disappeared in the favourite scripts list.	As expected	 	Pass

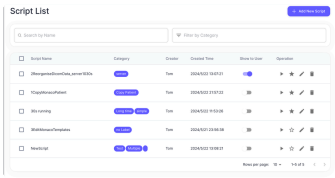
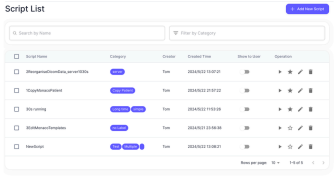
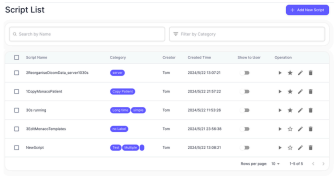
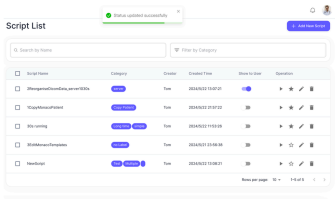
Sprint 3 Testing

- Test case 08
- Test case 09
- Test case 10
- Test case 11
- Test case 12

Test case 08

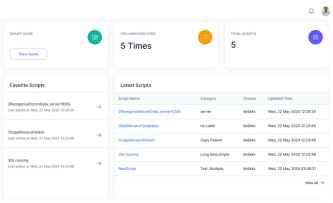
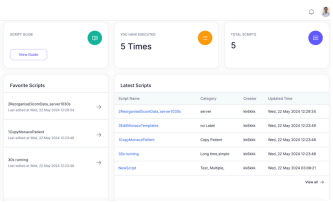
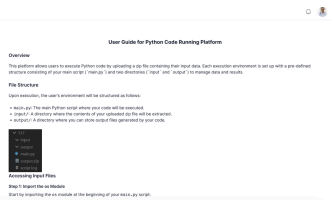
Test Case ID	8
Created By	@Yuncong Ji
Test Case Description	hide-active scripts
Tester's Name	@Yuncong Ji
Date Tested	May 22, 2024
Test Case (Pass/Fail/Not Executed)	Pass

Step #	Owner	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	@Yuncong Ji	Navigate to the “script list“ page	Show all scripts and its “show to user“ toggle switches	As expected		Pass
2	@Yuncong Ji	Click on the Toggle Switch	The Toggle Switch moves to the 'on' position and changes color to indicate it is 'on'.	As expected	before:  after: 	Pass
3	@Yuncong Ji	Click on the Toggle Switch.	The Toggle Switch moves to the 'off' position and	As expected	before:	Pass

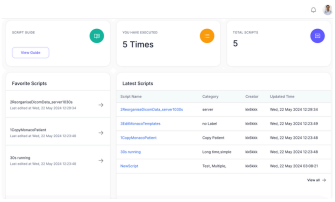
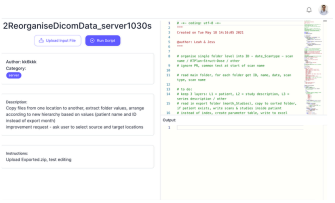
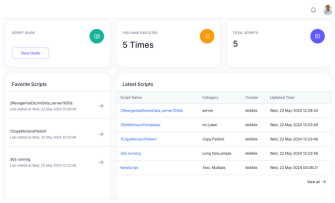
			changes color to indicate it is 'off'.		 after: 	
4	@Yuncong Ji	Click on the Toggle Switch.	When click switch, the notification appears with the message "Status updated successfully".	As expected	before:  after 	Pass

Test case 09

Test Case ID	9
Created By	@Yuncong Ji
Test Case Description	overview
Tester's Name	@Yuncong Ji
Date Tested	May 22, 2024
Test Case (Pass/Fail/Not Executed)	Pass

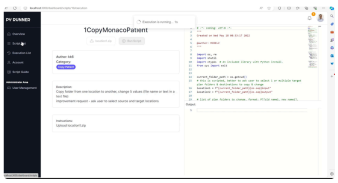
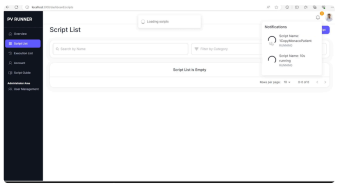
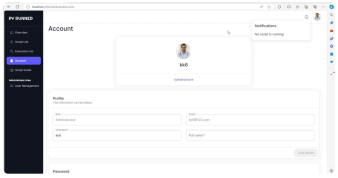
Step #	Owner	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	@Yuncong Ji	Navigate to the “Overview“ page	Show “YOU HAVE EXECUTED “ and the number of “TOTAL SCRIPTS”	As expected		Pass
2	@Yuncong Ji	Click the button “View Guide“	It will jump to page “Script guide” Page	As expected	before:  after: 	Pass

3	@Yuncong Ji	Click on "arrow button“ under Favorite Scripts	It will jump to page “Favorite Scripts” Page	As expected	before:	Pass

					 after: 	
4	@Yuncong Ji	Click on "link button“ under Latest Scripts	It will jump to page “Latest Scripts” Page	As expected	Before:  After: 	Pass

Test case 10

Test Case ID	10
Created By	@Zhihao Liang
Test Case Description	Notification
Tester's Name	@Zhihao Liang
Date Tested	May 22, 2024
Test Case (Pass/Fail/Not Executed)	Pass

Step #	Owner	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	@Zhihao Liang	Run a script on the script-running page	Show a new notification	As expected		Pass
2	@Zhihao Liang	Run another script on the script-running page	Show a new notification	As expected		
3	@Zhihao Liang	Wait until all running is finished	Notification disappear	As expected		

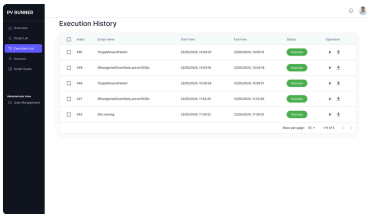
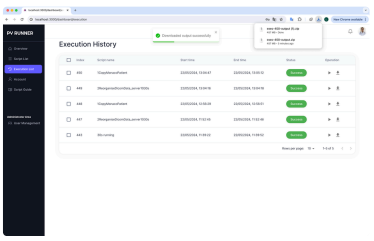
Test case 11

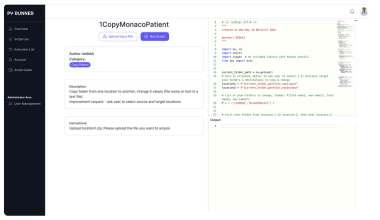
Test Case ID	11
Created By	@yijun liu
Test Case Description	The script guide page can be redirected by clicking on the home page.
Tester's Name	@yijun liu
Date Tested	May 22, 2024
Prerequisite	None
Test Case (Pass/Fail/Not Executed)	Pass

Step #	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:3000/dashboard	Click "View Guide" to jump to the script guide interface	As Expected		Pass

Test case 12

Test Case ID	12
Created By	@Chloe_Duan
Test Case Description	The execution history page to work as expected. 1. Add new execution history when user executed more scripts 2. User could access previous output history
Tester's Name	@Chloe_Duan
Date Tested	May 22, 2024
Prerequisite	None
Test Case (Pass/Fail/Not Executed)	Pass

Step #	Step Details	Expected Results	Actual Results	Screenshot	Pass / Fail / Not executed / Suspended
1	Navigate to http://localhost:3000/dashboard/execution	Click "Execution List" to jump to the script guide interface	As Expected		Pass
2	Navigate to http://localhost:3000/dashboard/execution	Click Download Button to dowloaded history output			Pass
3	Navigate to http://localhost:3000/dashboard/execution	Click Run Button to dowloaded history output			Pass

Code Review

This document outlines the code review process employed during our project. The review was conducted manually by team members, with each member participating in an average of 3-4 reviews. This document aims to standardize our approach to ensure consistency and quality across our codebase.

Objectives

- **Identify and resolve logical errors**: Ensuring the code functions correctly according to specifications.
- **Enforce coding standards**: Maintaining a consistent style and adhering to best practices to make the code more readable and maintainable.
- **Promote code simplicity**: Encouraging clear and concise code that is easy to read and understand.
- **Evaluate reusability**: Assessing the code for potential reuse to reduce redundancy and improve efficiency.

Process

1. **Initial Review**: Each team member is assigned code segments to review based on the criteria outlined.
2. **Feedback Submission**: Reviewers provide feedback and suggest improvements through inline comments or summary reports.
3. **Implementation of Changes**: Developers address the feedback by making the necessary revisions to their code.
4. **Follow-Up Review**: Revised code segments are submitted for a second review to ensure all concerns are addressed.
5. **Approval for Pull Request**: Once the code passes the second review, it is approved to be merged into the main branch via a pull request.

Review Criteria

- **Logical Errors**: Reviewers are tasked with identifying any flaws in logic that could lead to incorrect program behavior.
- **Code Standards**: Compliance with predefined coding guidelines is checked rigorously.
- **Simplicity**: Efforts are made to keep the codebase simple and understandable.
- **Reusability**: Code is evaluated on its potential for reuse in other parts of the project or future projects.

Sprint 2 Code Review

- Code Review For Authorization API
- Code Review for Scripts-related API
- Code Review for Execution Management Component
- Code Review for Script List
- Code Review for Log-in, API connection, Web UI integration
- Code Review for favourite API

Code Review For Authorization API

Project Name/Module Name: Authorization function

Author: Zhihao Liang

Reviewer: Yuncong Ji

Review Date: 13/3; 25/4

```python
import traceback

from flask import Blueprint, request
from flask_jwt_extended import jwt_required, get_current_user

from app.services.user_service import create_user, login, get_user_by_email, is_admin
import app.services.user_service as user_service
from app.utils.common import error_resp, success_response, log

auth_bp = Blueprint("auth_bp", __name__)


def admin_required(func):
    def wrapper(*args, **kwargs):
        current_user = get_current_user()
        if not is_admin(current_user['id']):
            return error_resp("Unauthorized operation", 401)
        return func(*args, **kwargs)

    return wrapper


@auth_bp.route("/api/users", methods=["POST"])
def register():
    # Registration logic here
    try:

        username = request.json["username"]
        email = request.json["email"]
        password = request.json["password"]
        is_normal = request.json["role"]
        user = create_user(username, email, password, is_normal)

        return success_response(
            message="User created successfully",
            data=user.get_claim(),
        )
    except Exception as e:
        traceback.print_exc()
        return error_resp(str(e), 401)


@auth_bp.route("/api/login", methods=["POST"])
def login2():
    # Login logic here
    try:
        email = request.json["email"]
```

```
        password = request.json["password"]
        user, token = login(email, password)
        if not user:
            return error_resp("Email or password is incorrect", 401)

        access_token = token
        log().info(f"User {user.email} logged in")
        return success_response(
            "Login successfully",
            {"access_token": access_token, "user": user.get_sanitized()},
        )
    except Exception as e:
        traceback.print_exc()
        return error_resp(str(e), 401)


@auth_bp.route("/api/current_user", methods=["GET"])
@jwt_required()
def current_user_info():
    try:
        current_user = get_current_user()
        log().info(f"Current user: {current_user}")
        user = get_user_by_email(current_user['email'])
        if user:
            return success_response(
                "User found",
                user.get_claim(),
            )
    except Exception as e:
        traceback.print_exc()
        return error_resp(str(e), 401)


@auth_bp.route("/api/users/<user_id>", methods=["DELETE"])
@jwt_required()
@admin_required
def del_user(user_id):
    if not user_service.delete_user(user_id):
        return error_resp("User not found", 404)
    user_service.delete_user(user_id)
    return success_response("User deleted successfully", {})


# TODO admin required
@auth_bp.route("/api/users", methods=["GET"])
def list_users():
    try:
        users = user_service.list_users()
        return success_response("Users found", users)
    except Exception as e:
        traceback.print_exc()
        return error_resp(str(e), 401)


@auth_bp.route("/api/users/<user_id>", methods=["GET"])
def get_user(user_id):
    try:
        user = user_service.get_user_by_id(user_id)
```

```
        return success_response("User found", user.get_sanitized())
    except Exception as e:
        traceback.print_exc()
        return error_resp(str(e), 401)
```

Code Review for First Version Code:

1. Exception Handling Consistency:

- The exception handling across the routes is inconsistent. While some routes print the exception trace, others do not. Consistent error handling ensures easier debugging and maintenance.

2. Security Implications:

- The deletion route (`@auth_bp.route("/api/users/<user_id>", methods=["DELETE"])`) calls `delete_user` twice if the user exists. This could lead to potential unintended side effects or performance issues.

3. Code Duplication:

- The `admin_required` decorator and various routes use similar checks and patterns repeatedly, which could be simplified or made more DRY (Don't Repeat Yourself).

4. Logging:

- Logging within routes does not consistently capture significant events, especially around authentication failures or system critical changes, such as user deletion or role changes.

5. Use of HTTP Status Codes:

- The usage of HTTP status code `401` (Unauthorized) is often misused. For example, when a user is not found during deletion, it returns `404` (Not Found), which is correct, but for other errors, `401` is used regardless of the nature of the error. The correct status code should reflect the specific error encountered.

Next Actions:

1. **Standardize Exception Handling:** Implement a unified approach to handle exceptions and logging throughout the application. This can involve creating a helper function or a more robust error handling mechanism.
2. **Review Security Practices:** Ensure that operations such as deleting a user are idempotent and do not repeat unnecessary database calls.
3. **DRY Code Practices:** Look into refactoring common functionality like user checks into reusable components or middleware.
4. **Enhance Logging:** Implement more comprehensive logging especially for critical operations like logins, user creation, and user deletion.
5. **Correct HTTP Status Codes:** Review and apply the correct HTTP status codes based on the error type, improving the API's response accuracy to client requests.

```
from flask import Blueprint, request
from flask_jwt_extended import jwt_required, get_current_user

from app.services.user_service import (
    create_user,
    login,
    get_user_by_email,
    is_admin,
)
import app.services.user_service as user_service
from app.utils.common import error_resp, success_response, log, current_user
import base64


auth_bp = Blueprint("auth_bp", __name__)
```

```python


def admin_required(func):
    def wrapper(*args, **kwargs):
        current_user = get_current_user()
        if not is_admin(current_user["id"]):
            return error_resp("Unauthorized operation", 401)
        return func(*args, **kwargs)

    return wrapper


@auth_bp.route("/api/users", methods=["POST"])
def register():
    # Registration logic here
    username = request.json["username"]
    email = request.json["email"]
    password = request.json["password"]
    is_normal = request.json["role"]
    user = create_user(username, email, password, is_normal)

    return success_response(
        message="User created successfully",
        data=user.get_claim(),
    )


@auth_bp.route("/api/login", methods=["POST"])
def login2():
    # Login logic here
    email = request.json["email"]
    password = request.json["password"]
    user, token = login(email, password)
    if not user:
        return error_resp("Email or password is incorrect", 401)

    access_token = token
    log().info(f"User {user.email} logged in")
    return success_response(
        "Login successfully",
        {"access_token": access_token, "user": user.get_claim()},
    )


@auth_bp.route("/api/current_user", methods=["GET"])
@jwt_required()
def current_user_info():
    current_user = get_current_user()
    log().info(f"Current user: {current_user}")
    user = get_user_by_email(current_user["email"])
    if user:
        return success_response(
            "User found",
            user.get_claim_with_avatar(),
        )


@auth_bp.route("/api/users/<user_id>", methods=["DELETE"])
```

```python
@jwt_required()
@admin_required
def del_user(user_id):
    if not user_service.delete_user(user_id):
        return error_resp("User not found", 404)
    user_service.delete_user(user_id)
    return success_response("User deleted successfully", {})


# TODO admin required
@auth_bp.route("/api/users", methods=["GET"])
def list_users():
    users = user_service.list_users()
    return success_response("Users found", users)


@auth_bp.route("/api/users/<user_id>", methods=["GET"])
def get_user(user_id):
    user = user_service.get_user_by_id(user_id)
    if not user:
        return error_resp("User not found", 404)
    return success_response("User found", user.get_sanitized())


@auth_bp.route("/api/users", methods=["PUT"])
@jwt_required()
def update_user():

    email = request.json["email"]
    c_user = current_user()
    if c_user.email != email and not is_admin(c_user.id):
        return error_resp("Unauthorized operation", 401)
    update_password = False
    new_vals = {}
    if "username" in request.json:
        new_vals["username"] = request.json["username"]
    if "password" in request.json:
        update_password = True
        new_vals["password"] = request.json["password"]
    if "avatar" in request.json:
        new_vals["avatar"] = base64.b64decode(request.json["avatar"])
    if "role" in request.json:
        new_vals["role_id"] = request.json["role"]
    log().info(f"Updating user {email}")
    user = user_service.update_user(email, new_vals, update_password)
    return success_response("User updated successfully", user.get_sanitized())

```

Code Review for Second Version Code:

1. Improvements in Error Handling and Logging:

- The new version includes a more streamlined approach to logging important events, which can greatly aid in system monitoring and debugging.

2. Functionality Enhancements:

- Introduction of `update_user` route adds functionality for user profile updates, which is a positive improvement addressing user management comprehensively.

3. Avatar Handling:

- The addition of avatar handling via Base64 encoding could introduce issues related to data size and validation. Ensuring that the avatar data does not exceed acceptable sizes should be considered.

4. Security and Permissions Checks:

- Improved checks on role and email changes in the `update_user` route, enhancing security by preventing unauthorized role elevation or profile changes.

Next Actions:

1. **Data Validation:** Implement thorough input validation, especially for encoded data like avatars to prevent common vulnerabilities (e.g., buffer overflow or injection attacks).
2. **Optimize User Deletion:** Ensure that the `delete_user` function is called once and handle potential errors or non-existent users gracefully.
3. **Enhance Role Handling:** Consider adding explicit routes or utilities for managing user roles, especially for listing administrators as indicated by the TODO comments.
4. **Refine Error Responses:** Refine error messaging and status codes to provide more specific feedback for client-side error handling.

Code Review for Scripts-related API

Project Name/Module Name: Scripts-related APIs and services

Author: @Yuncong Ji

Reviewer: @Lingyi Kong

Review Date: Apr 23, 2024 , Apr 30, 2024

Code Review for First Version

Selection Criteria:

- Priority was given to reviewing new features added in the sprint.
- Areas with high bug rates in previous releases were targeted.

Process:

- The review was initiated by examining the commits made last week, focusing on modules related to script management.
- The session was conducted remotely via a scheduled Zoom call, where each line of code was shared and discussed live.
- Feedback was documented directly in the code using comments and later summarized on this Confluence page.

Feedback Received:

1. Exception Handling: Inconsistencies in how exceptions are caught and handled.
2. Security Concerns: Potential security risks due to improper validation of script deletion logic.
3. Code Optimization: Unnecessary duplication in the codebase that could be simplified using helper functions.

Actions Taken:

- Exception Handling: Standardized the exception handling by introducing a base class for exceptions and consistent methods for their management across all modules.
- Security Improvements: Added checks and validation for script deletion functions to prevent unauthorized access and ensure data integrity.
- Code Refactoring: Common functionalities were abstracted into utility functions to reduce redundancy and improve maintainability.

Code Changes:

- Added a new base exception class.
- Implemented new validation logic for script deletions.
- Refactored redundant code patterns into shared utility functions.

Unchanged Aspects:

- Some suggested optimizations were deferred to the next iteration due to time constraints.

Code Review for Second Version

Selection Criteria:

- Code associated with newly introduced features and critical bug fixes.
- Modules where significant refactoring was done in the previous iterations.

Process:

- Reviewed changes made in response to feedback from the first review.
- Used a shared Git repository to highlight changes, facilitating easier discussion and annotation.

Feedback Received:

1. Data Handling: Concerns over handling large data sets with the new script functionalities.
2. Error Handling: Need for more detailed client-side error messages for better user experience.
3. Performance Issues: Potential slow-downs due to inefficient database queries in script management.

Actions Taken:

- Data Handling: Implemented pagination and optimized queries to handle large data sets efficiently.
- Error Handling: Enhanced the granularity of error responses to provide more specific feedback to the client.
- Performance Optimization: Refactored database access layers to improve performance and scalability.

Code Changes:

- Introduced pagination for listing scripts.
- Enhanced error handling mechanisms.
- Optimized database queries to improve response times.

Unchanged Aspects:

- Certain error handling improvements were postponed to future reviews for further assessment and testing.

Code Review for Execution Management Component

Project Name/Module Name: Execution Management

Author: @Junhao KONG

Reviewer: @yijun liu

Date: Apr 27, 2024 May 31, 2024

Tools Used: Manual inspection, VS Code for syntax highlighting.

Code Review for the First Version:

1. **File Upload Handling:**
   - Issue: There is no error handling if the file upload fails or is in an incorrect format.
   - Suggestion: Implement error handling and feedback to the user for file uploads.
2. **File Download and Command Execution:**
   - Issue: The functions for downloading files and executing commands are placeholders without actual implementation.
   - Suggestion: Define the backend API or logic for downloading files and executing commands. Consider security implications, especially with command executions.
3. **Accessibility and Usability:**
   - Issue: Input fields lack accessibility features such as ARIA labels.
   - Suggestion: Add `aria-label` attributes to improve accessibility.
4. **React Component Optimization:**
   - Issue: Potential re-renders due to anonymous functions in the render method.
   - Suggestion: Convert these anonymous functions to named functions outside the component body or use `useCallback` to prevent unnecessary re-renders.

Next Actions:

1. Refactor Icon Imports: Change SSR icons to client-side icons.
2. Implement File Type Validation: Specify acceptable file types using the `accept` attribute on the file input.
3. Improve Error Handling: Implement and improve existing error handling mechanisms.
4. Enhance Accessibility: Add accessibility features like `aria-labels`.
5. Optimize Handler Functions: Refactor handlers using `useCallback` to improve performance.

Code Review for the Second Version：

1. **Code Optimization:**
   - Issue: Inline function definitions in the render method, such as those in the `onChange` props of the `TextField` components, can lead to performance issues due to re-rendering.
   - Suggestion: Use `useCallback` for handlers to prevent unnecessary re-renders. Define these handlers outside of the component or memoize them.
2. **Security Concerns**:
   - Issue: Direct execution of commands or file handling without sanitization or validation poses a security risk.
   - Suggestion: Implement validation for the file types and sizes on the client side and especially on the server side. Ensure commands executed are sanitized to prevent injection attacks.

Next Actions:

1. Enhance Security Measures: Add validation and sanitization for file handling and command execution.
2. Optimize Handler Functions: Refactor handlers using `useCallback` to improve performance.
3. Standardize Styling Approach: Transition to a more maintainable styling methodology like CSS modules.

Code Review for Script List

Project Name/Module Name: Execution Management

Author: @yijun liu

Reviewer: @Junhao KONG

Date: Apr 27, 2024 May 31, 2024

Tools Used: Manual inspection, VS Code for syntax highlighting.

Code Review for the First Version of Code

General Observations

- The code lacks concise comments, which can make it hard for others to understand the purpose and functionality of each section or function.
- There are some commented-out code blocks (e.g., buttons for importing and exporting) that either should be removed or clearly explained why they are retained.

Specific Issues and Suggestions

1. React Component Structure

Issue: The usage of inline functions in JSX props can cause performance issues due to unnecessary re-renders. **Suggestion:** Convert these inline functions to named functions outside the component body or use `useCallback` to wrap these functions, which can prevent them from being recreated on every render.

2. State Management

Issue: The management of state and side-effects can be optimized. The functions and state updates seem scattered and could be more organized. **Suggestion:** Consider using `useReducer` for complex state logic or moving related state handling into custom hooks to clean up the main component body.

3. Accessibility and Usability

Issue: Same as previously identified, input fields lack proper accessibility features. **Suggestion:** Add `aria-label` attributes to input fields and buttons to improve accessibility. Ensure all interactive elements are accessible via keyboard navigation.

4. Error Handling

Issue: There is a lack of error handling around state updates and asynchronous operations. **Suggestion:** Implement robust error handling, especially in asynchronous functions like `handleAddScrip`. Use try/catch blocks and provide user feedback on errors.

5. File Handling

Issue: Icons are imported from SSR-specific files, which might not be optimal for client-side rendering. **Suggestion:** Change the imports for icons from SSR to client-specific modules to avoid potential rendering issues on the client side.

Code Review for the Second Version of Code

General Observations

- Better organization of code with proper modularization of functions and separation of concerns.

- Usage of hooks like `useEffect` for fetching data and managing side effects shows good practice in managing lifecycle events in functional components.

Specific Issues and Suggestions

1. Security and Validation

Issue: Direct execution of scripts or commands without proper validation or sanitization. **Suggestion:** Ensure all file and command executions are properly sanitized to prevent security vulnerabilities such as SQL injection or XSS. Validate all inputs both on the client and server sides.

2. Performance Optimization

Issue: Inline function definitions still persist in the second version, which can lead to the same performance issues. **Suggestion:** Use `useCallback` for all handlers and define these handlers outside of the component or memoize them to prevent unnecessary re-renders.

3. Error Handling and Feedback

Issue: The current implementation does not adequately handle errors or provide feedback for operations like deleting or favoriting scripts. **Suggestion:** Implement more comprehensive error handling and provide clear user feedback for operations. Use mechanisms like toasts or dialog boxes to inform the user of the status of their actions.

4.Accessibility Improvements

Issue: Continued lack of comprehensive accessibility features. **Suggestion:** Expand the use of accessibility features beyond `aria-labels`. Consider implementing keyboard navigation support and focus management to enhance usability for all users.

Next Steps

- **Enhance Security Measures:** Add more robust validation and sanitization for file handling and command execution.
- **Optimize Handler Functions:** Continue refactoring handlers using `useCallback` to improve performance..

Code Review for Log-in, API connection, Web UI integration

Project Name/Module Name: Log-in, API connection, Web UI integration

Author: @Chloe_Duan

Reviewer: @Zhihao Liang

Date: Apr 23, 2024 Apr 27, 2024 May 1, 2024

Tools Used: Manual inspection, VS Code for syntax highlighting.

1. Log-in Component Review

Code Snippet for Review:

```
  const onSubmitRefactor = React.useCallback(
  async (values: Values): Promise<void> => {
    setIsPending(true);
    try {
      const result = await authClient.signInWithPassword(values);

      // If error happens when connecting API
      if ('error' in result) {
        setError('root', { type: 'server', message: result.error });
        setIsPending(false);
        return;
      }

      await checkSession?.();

      // UserProvider, for this case, will not refresh the router
      // After refresh, GuestGuard will handle the redirect
      router.refresh();
    } catch (error) {
      console.error('Error during login:', error);
      setError('root', { type: 'server', message: 'An unexpected error occurred' });
    } finally {
      setIsPending(false);
    }
    router.refresh();
  },
  [checkSession, router, setError]
);

async signInWithPassword(params: SignInWithPasswordParams): Promise<ApiStatusResponse> {
  try {
    const response = await login(params);

    if (!response.success) {
      return { error: response.message || 'Login failed' };
    }

    // console.log('Login successful:', response.data);
    if (!response.data.user) {
      return { error: 'User data is missing' };
    }
```

```
      const user = parseUser(response.data);
      localStorage.setItem('custom-auth-token', response.data.access_token);
      return user;
    } catch (error) {
      console.error('Error during signInWithPassword:', error);
      return { error: 'Email or password is incorrect.' };
    }
```

Review Points:

- **Logical Errors:** Check if the condition to determine when to send the request is adequate. Should we check for just empty strings, or also null/undefined values?
- **Code Standard Compliance:** Ensure naming conventions are followed and that the code adheres to ES6+ standards if applicable. Is the fetch API used correctly?
- **Code Simplicity:** Is the error handling strategy adequate? Could it be simplified or made more robust?
- **Reusability:** The function could be more reusable if it accepted a callback or returned a promise that resolves or rejects based on the authentication success.

Next Steps After Review:

- **Pull Request:** Incorporate any accepted changes based on feedback and prepare for a merge after final approval.
- **Feedback Integration:** Address all comments critically, improving error handling, perhaps by adding specific error messages for the user based on different failed states (invalid credentials, server error, etc.).

2. API Connection Component Review

Code Snippet for Review:

```
  export interface ListUsersResponse {
  avatar?: string;
  created_at?: string;
  created_by?: string;
  email: string;
  id: number;
  last_login?: string;
  role_id: number;
  updated_at?: string;
  username: string;
}

// list users
export async function listUsers(): Promise<APIResponse<ListUsersResponse[]>> {
  return request<ListUsersResponse[]>({
    url: '/api/users',
    method: 'GET',
  });
}
  useEffect(() => {
    const loadData = async () => {
      await fetchUsers();
    };
    toast.loading('Loading users...');
    loadData()
      .then(() => {
        toast.success('Users loaded successfully');
      })
      .catch((e) => {
        console.log(e);
```

```
      })
      .finally(() => {
        toast.dismiss();
      });
  }, []);

  const fetchUsers = async () => {
    try {
      const response = await listUsers();
      setUsers(response.data.map(parseUser));
    } catch (error) {
      console.error('Failed to fetch users:', error);
    }
  };
```

Review Points:

- **Logical Errors:** Ensure proper error handling is implemented.
- **Code Standard Compliance:** Is async/await used properly? Are there any potential unhandled promise rejections?
- **Code Simplicity:** Is the function handling more than it should? Could the error handling be more specific based on the error codes?
- **Reusability:** This function is quite reusable; however, parameterizing the error message might make it more versatile.

Next Steps After Review:

- **Pull Request:** Update the function to handle different types of HTTP status codes appropriately.
- **Feedback Integration:** Modify the function based on received feedback to handle specific HTTP errors and possibly retry logic for specific failures.

3. Web UI Integration Component Review

Code Snippet for Review:

```
import { getSiteURL } from '@/lib/get-site-url';
import { LogLevel } from '@/lib/logger';

export interface Config {
  site: { name: string; description: string; themeColor: string; url: string };
  logLevel: keyof typeof LogLevel;
}

export const config: Config = {
  site: { name: 'Py Runner', description: '', themeColor: '#090a0b', url: getSiteURL() },
  logLevel: (process.env.NEXT_PUBLIC_LOG_LEVEL as keyof typeof LogLevel) ?? LogLevel.ALL,
};

import type { NavItemConfig } from '@/types/nav';
import { paths } from '@/paths';

export const navAdminItems = [
  { key: 'user', title: 'User Management', href: paths.dashboard.customers},
] satisfies NavItemConfig[];

export const navItems = [
  { key: 'overview', title: 'Overview', href: paths.dashboard.overview},
  { key: 'script', title: 'Script List', href: paths.dashboard.scripts},
  { key: 'account', title: 'Account', href: paths.dashboard.account},

] satisfies NavItemConfig[];
```

Review Points:

- **Logical Errors:** Check if DOM elements exist before adding event listeners.
- **Code Standard Compliance:** Use modern DOM manipulation techniques if applicable.
- **Code Simplicity:** Could these methods be simplified or broken down into smaller parts?
- **Reusability:** Is there a way to make this component more modular?

Next Steps After Review:

- **Pull Request:** Refactor the UI class to use more modular methods and modern JavaScript features.
- **Feedback Integration:** Adjust based on feedback, possibly changing how DOM elements are managed or introducing a state management system.

In each case, after incorporating the feedback, further tests should be conducted to ensure that the changes are effective and do not introduce new issues. Depending on the feedback's nature, not all suggestions might be implemented, particularly if they conflict with other design considerations or project constraints.

Code Review for favourite API

Module Name: Favorite function

Author: Lingyi Kong @Lingyi Kong

Reviewer: Yuncong Ji @Yuncong Ji

Review Date: 23/4

Code Review

1. Code Structure and Naming Convention:

- [x] Code structure appears clear and organized.
- [ ] Function and variable names are descriptive and adhere to the naming conventions.

2. Functionality Implementation:

- [x] `add_favorite`: Validates if the script is already in favorites or if the script exists before adding it.
- [x] `delete_favorite`: Properly handles the case where the favorite to be deleted doesn't exist.
- [x] `list_favorite`: Checks if the user exists before listing favorites.

3. Comments and Documentation:

- [x] Add comments/docstrings explaining the purpose of functions and any complex logic, especially in utility functions like `add_favorite`, `delete_favorite`, and `list_favorite`.

4. Test Coverage:

- [ ] Include unit tests to cover different scenarios for `add_favorite`, `delete_favorite`, and `list_favorite` functions.

5. Performance and Security:

- [ ] Ensure that sensitive operations like adding/deleting favorites are protected with JWT authentication.

6. Maintainability:

- [x] Consolidate database session management into a reusable function to reduce code duplication.
- [ ] Use constants for HTTP status codes and error messages to enhance maintainability and consistency.
- [ ] Evaluate potential improvements for error handling, such as custom exceptions for clearer error propagation.

Next review action: 28/4

Additional Suggestions:

- Consider adding validation for `script_id` parameter in API endpoints to ensure it's a positive integer.
- Refactor repetitive code, such as database session management, into utility functions or decorators for cleaner codebase maintenance.

Module Name: Favorite function

Author: Lingyi Kong @Lingyi Kong

Reviewer: Yuncong Ji @Yuncong Ji

Review Date: 28/4

Code Review

1. Code Structure and Naming Convention:

- [x] Code structure appears clear and organized.
- [x] Function and variable names are descriptive and adhere to the naming conventions.

2. Functionality Implementation:

- [x] `add_favorite`: Validates if the script is already in favorites or if the script exists before adding it.
- [x] `delete_favorite`: Properly handles the case where the favorite to be deleted doesn't exist.
- [x] `list_favorite`: Checks if the user exists before listing favorites.

3. Comments and Documentation:

- [x] Add comments/docstrings explaining the purpose of functions and any complex logic, especially in utility functions like `add_favorite`, `delete_favorite`, and `list_favorite`.

4. Test Coverage:

- [x] Include unit tests to cover different scenarios for `add_favorite`, `delete_favorite`, and `list_favorite` functions.

5. Performance and Security:

- [x] Ensure that sensitive operations like adding/deleting favorites are protected with JWT authentication.

6. Maintainability:

- [x] Consolidate database session management into a reusable function to reduce code duplication.
- [x] Use constants for HTTP status codes and error messages to enhance maintainability and consistency.
- [x] Evaluate potential improvements for error handling, such as custom exceptions for clearer error propagation.

Sprint 3 Code Review

- Code Review for Asynchronous Tasks Handling API
- Code Review for Visible/Invisible
- Code Review for Overview Page
- Code Review for the Active/ Inactive, Script-related APIs
- Code Review for upload and complete api

Code Review for Asynchronous Tasks Handling API

Project Name/Module Name: Asynchronous Tasks Handling

Author: Zhihao Liang

Reviewer: Yuncong Ji

Review Date: 15/5; 20/5

```python
import json
import os
import time

from celery.result import AsyncResult
from flask import Blueprint, request
from flask import Response
from flask import current_app, send_file
from flask_jwt_extended import jwt_required

import app.services.execution_service as execution_service
from app.services.execution_service import execute_script, save_tmp_file
from app.utils.common import error_resp
from app.utils.common import success_response, get_current_user_id

exec_bp = Blueprint("exec_bp", __name__)


@exec_bp.route("/api/execute", methods=["POST"])
@jwt_required()
def execution():
    # get user id from jwt
    user_id = get_current_user_id()
    if "multipart/form-data" not in request.content_type:
        return error_resp("Invalid content type, must be multipart/form-data", 400)
    script_id = request.form.get("script_id", "")
    has_file = request.form.get("has_file", False)
    filename = None
    if "file" in request.files:
        filename = save_tmp_file(user_id, request.files["file"])
    new_exec = execute_script(int(script_id), user_id, input_zip_path=filename)
    return success_response("Execution started successfully", new_exec)


@exec_bp.route("/api/download/<string:execution_id>", methods=["GET"])
def download_file(execution_id):
    if not execution_id:
        return error_resp("No filename specified", 400)
    base_path = current_app.config["EXECUTION_DIR"]
    return send_file(
        f"{base_path}{os.sep}{execution_id}{os.sep}output.zip", as_attachment=True
    )


@exec_bp.route("/api/download/<string:execution_id>/log", methods=["GET"])
def download_log(execution_id):
    if not execution_id:
```

```
        return error_resp("No filename specified", 400)
    base_path = current_app.config["EXECUTION_DIR"]
    return send_file(
        f"{base_path}{os.sep}{execution_id}{os.sep}script.log", as_attachment=True
    )




def sse_response(data):
    return f"data: {json.dumps(data)}\n\n"


def stream(task_id):
    res = AsyncResult(task_id)

    def generate():
        if res is None:
            yield sse_response({"status": "error", "message": "Task not found"})
            return
        # Yield initial data
        while not res.ready():
            time.sleep(1)
            yield sse_response({"status": res.state, "progress": res.info})
        result = json.dumps(res.info)
        yield sse_response({"status": "finished", "result": result})

    return Response(generate(), mimetype="text/event-stream")


@exec_bp.route("/api/v2/execute", methods=["POST"])
def execution2():
    # get user id from jwt
    user_id = get_current_user_id()
    if "multipart/form-data" not in request.content_type:
        return error_resp("Invalid content type, must be multipart/form-data", 400)
    script_id = request.form.get("script_id", "")
    script_id = int(script_id)
    has_file = request.form.get("has_file", False)
    filename = None
    if "file" in request.files:
        filename = save_tmp_file(user_id, request.files["file"])
    new_exec_id, task_id = execution_service.init_exec(script_id, user_id, filename)
    return success_response(
        "Execution started successfully", {"exec_id": new_exec_id, "task_id": task_id}
    )


@exec_bp.route("/api/execute/status", methods=["POST"])
def execute():
    # Assume the POST body contains the script to execute.
    id = request.json.get("taskId")
    return stream(id)
```

Code Review for First Version Code:

1. **Exception Handling Consistency:**

- The code lacks consistent exception handling. For example, if `execute_script` or `save_tmp_file` fail, there are no try-except blocks to catch and log these exceptions.
- Recommendation: Wrap critical operations in try-except blocks and log the exceptions to ensure easier debugging and maintenance.

2. **Security Implications:**
   - The `execution` and `execution2` endpoints do not check if the user has the right permissions to execute the script.
   - The `download_file` and `download_log` endpoints do not verify the user's authorization to download the specified files. This can lead to unauthorized access to files.
   - Recommendation: Implement permission checks for executing scripts and downloading files/logs.
3. **Code Duplication:**
   - The code for checking the content type and saving the file is duplicated in both `execution` and `execution2` endpoints.
   - Recommendation: Refactor the common logic into a reusable function to follow DRY (Don't Repeat Yourself) principles.
4. **Logging:**
   - The current implementation lacks logging for critical operations such as starting an execution, saving files, and handling errors.
   - Recommendation: Add logging statements to capture significant events and errors, which will help in monitoring and debugging.
5. **Use of HTTP Status Codes:**
   - The `error_resp` function should use more specific HTTP status codes instead of defaulting to 400 for all errors. For example, if a file is not found, a 404 status code should be returned.
   - Recommendation: Review and apply the correct HTTP status codes based on the specific errors encountered.

Next Actions:

1. **Standardize Exception Handling:**
   - Implement a unified approach to handle exceptions and logging throughout the application, such as using a decorator for error handling.
2. **Review Security Practices:**
   - Ensure that operations such as executing a script or downloading files are properly authorized. Implement permission checks for these operations.
3. **DRY Code Practices:**
   - Refactor common logic into reusable functions to reduce code duplication. For example, the file upload handling can be moved to a separate helper function.
4. **Enhance Logging:**
   - Add logging for critical operations, such as starting an execution, saving files, and handling errors. This will help in monitoring and debugging.
5. **Correct HTTP Status Codes:**
   - Review and apply the correct HTTP status codes based on the specific errors encountered. Ensure that the API responses accurately reflect the nature of the errors.

```python
import json
import os
import time

from celery.result import AsyncResult
from flask import Blueprint, request
from flask import Response
from flask import current_app, send_file
from flask_jwt_extended import jwt_required

import app.services.execution_service as execution_service
from app.services.execution_service import execute_script, save_tmp_file
from app.utils.common import error_resp
from app.utils.common import success_response, get_current_user_id
```

```
from typing import List
exec_bp = Blueprint("exec_bp", __name__)


@exec_bp.route("/api/execute", methods=["POST"])
@jwt_required()
def execution():
    # get user id from jwt
    user_id = get_current_user_id()
    if "multipart/form-data" not in request.content_type:
        return error_resp("Invalid content type, must be multipart/form-data", 400)
    script_id = request.form.get("script_id", "")
    has_file = request.form.get("has_file", False)
    filename = None
    if "file" in request.files:
        filename = save_tmp_file(user_id, request.files["file"])
    new_exec = execute_script(int(script_id), user_id, input_zip_path=filename)
    return success_response("Execution started successfully", new_exec)


@exec_bp.route("/api/download/<string:execution_id>", methods=["GET"])
def download_file(execution_id):
    if not execution_id:
        return error_resp("No filename specified", 400)
    base_path = current_app.config["EXECUTION_DIR"]
    return send_file(
        f"{base_path}{os.sep}{execution_id}{os.sep}output.zip", as_attachment=True
    )


@exec_bp.route("/api/download/<string:execution_id>/log", methods=["GET"])
def download_log(execution_id):
    if not execution_id:
        return error_resp("No filename specified", 400)
    base_path = current_app.config["EXECUTION_DIR"]
    return send_file(
        f"{base_path}{os.sep}{execution_id}{os.sep}script.log", as_attachment=True
    )


def sse_response(is_running, status, message, data=None)->str:
    return (
        f"data: {json.dumps({"is_running": is_running, "status": status, "message": message, "data": data}    )
    )









@exec_bp.route("/api/v2/execute", methods=["POST"])
def execution_v2():
    # get user id from jwt
    user_id = get_current_user_id()
    if "multipart/form-data" not in request.content_type:
        return error_resp("Invalid content type, must be multipart/form-data", 400)
```

```
    script_id = request.form.get("script_id", "")
    script_id = int(script_id)
    has_file = request.form.get("has_file", False)
    filename = None
    if "file" in request.files:
        filename = save_tmp_file(user_id, request.files["file"])
    new_exec_id, task_id = execution_service.init_exec(script_id, user_id, filename)
    return success_response(
        "Execution started successfully", {"exec_id": new_exec_id, "task_id": task_id}
    )


@exec_bp.route("/api/execute/status/all", methods=["GET"])
def sse_execute_status_all():
    user_id = request.json.get("userId")
    exec_list = execution_service.get_executions_by_status_user_id(user_id, "RUNNING")


    def exec_status_stream_list(exec_list:List[dict]):
        res_list= [AsyncResult(exec_["task_id"]) for exec_ in exec_list]

        def generate():
                # Yield initial data
            info=[{"script_id":exec_["script_id"],"script_name":exec_["script_name"],"task_id":exec_["task_id"]
            yield sse_response(True, "Listening", f"{len(exec_list)} executions are running",info)
            while True:
                time.sleep(1)
                running=[res for res in res_list if res.state=="RUNNING" or res.state=="PENDING"]
                success=[res for res in res_list if res.state=="SUCCESS"]
                failed=[res for res in res_list if res.state=="FAILED"]
                running_task_ids=[res.id for res in running]
                success_task_ids=[res.id for res in success]
                failed_task_ids=[res.id for res in failed]
                yield sse_response(True, "Listening", f"{len(running)} executions are running",{"running":runni
                if len(running)==0:
                    yield sse_response(False, "Listening", f"All executions are done",{"running":running_task_i
                    break

        return Response(generate(), mimetype="text/event-stream")
    return exec_status_stream_list(exec_list)

@exec_bp.route("/api/execute/status", methods=["GET"])
def sse_execute_status():
    # Assume the POST body contains the script to execute.
    id = request.args.get("taskId")
    def exec_status_stream(task_id):
        res = AsyncResult(task_id)

        def generate():
            if res is None:
                yield sse_response(False, "ERROR", "Task not found", None)
                return
            # Yield initial data
            while not res.ready():
                time.sleep(1)
                if res.state == "RUNNING":
                    yield sse_response(True, res.state, "Execution is running", None)
                # pending state means the task is not yet started
```

```
            yield sse_response (False, res.state, "Execution is done", res.info)

        return Response(generate(), mimetype="text/event-stream")
    return exec_status_stream(id)


EXEC_STATUS = ["PENDING", "RUNNING", "SUCCESS", "FAILED", "ABORTED"]


@exec_bp.route("/api/execute", methods=["GET"])
def get_exec_by_user_status():
    user_id = get_current_user_id()
    status = request.args.get("status")
    if status not in EXEC_STATUS:
        return error_resp(f"Status should be any of {EXEC_STATUS}", 400)

    return success_response(
        "Executions retrieved successfully",
        execution_service.get_executions_by_status_user_id(user_id, status),
    )
```

Code Review for Second Version Code:

1. **Exception Handling Consistency:**
   - The code lacks consistent exception handling. There are no try-except blocks around critical operations such as `execute_script`, `save_tmp_file`, and other service calls.
   - **Recommendation:** Wrap critical operations in try-except blocks and log exceptions to ensure easier debugging and maintenance.
2. **Security Implications:**
   - The `execution` and `execution_v2` endpoints do not check if the user has the right permissions to execute the script.
   - The `download_file` and `download_log` endpoints do not verify the user's authorization to download the specified files.
   - **Recommendation:** Implement permission checks for executing scripts and downloading files/logs.
3. **Code Duplication:**
   - The code for checking the content type and saving the file is duplicated in both `execution` and `execution_v2` endpoints.
   - **Recommendation:** Refactor the common logic into a reusable function to follow DRY (Don't Repeat Yourself) principles.
4. **Logging:**
   - The current implementation lacks logging for critical operations such as starting an execution, saving files, and handling errors.
   - **Recommendation:** Add logging statements to capture significant events and errors, which will help in monitoring and debugging.
5. **Use of HTTP Status Codes:**
   - The `error_resp` function should use more specific HTTP status codes instead of defaulting to 400 for all errors. For example, if a file is not found, a 404 status code should be returned.
   - **Recommendation:** Review and apply the correct HTTP status codes based on the specific errors encountered.

Next Actions:

1. **Standardize Exception Handling:**
   - Implement a unified approach to handle exceptions and logging throughout the application, such as using a decorator for error handling.
2. **Review Security Practices:**
   - Ensure that operations such as executing a script or downloading files are properly authorized. Implement permission checks for these operations.
3. **DRY Code Practices:**

- Refactor common logic into reusable functions to reduce code duplication. For example, the file upload handling can be moved to a separate helper function.

4. **Enhance Logging:**
   - Add logging for critical operations, such as starting an execution, saving files, and handling errors. This will help in monitoring and debugging.
5. **Correct HTTP Status Codes:**
   - Review and apply the correct HTTP status codes based on the specific errors encountered. Ensure that the API responses accurately reflect the nature of the errors.

Code Review for Visible/Invisible

Project Name/Module Name: Execution Management

Author: @Junhao KONG

Reviewer: @Junhao KONG

Date: May 15, 2024 May 22, 2024

Tools Used: Manual inspection, VS Code for syntax highlighting.

First Version Review

1. **No Empty State Handling:**
   - The code does not handle the case when the `scripts` array is empty, which can lead to a poor user experience.
2. **Redundant Code for Favorite Icon:**
   - The check for favorite scripts and rendering the appropriate icon can be simplified to reduce redundancy.
3. **Missing Status Toggle Feature:**
   - There is no feature to toggle the status of the scripts, which could be useful for admin users.
4. **TypeScript Types:**
   - The TypeScript types for the script objects and props could be more explicitly defined for better type safety and clarity.
5. **Error Handling:**
   - The code lacks error handling, especially in functions that involve asynchronous operations such as `onExecution`, `onFavScript`, `onDeleteScript`, and `onEditScript`.

First Version - Improvement Methods:

1. **Implement Empty State Display:**
   - Introduce logic to show a message if the `scripts` array is empty to improve user experience when there are no scripts to display.
2. **Streamline Favorite Icon Logic:**
   - Simplify the conditional rendering for the favorite icon to avoid redundancy by using a single ternary operator.
3. **Add Status Toggle Functionality:**
   - Implement a feature to toggle the status of scripts using a `Switch` component, enabling admins to change the status of scripts directly from the table.
4. **Define Comprehensive TypeScript Types:**
   - Ensure explicit TypeScript types for script objects and props to enhance type safety and improve code clarity.
5. **Introduce Error Handling:**
   - Add error handling in functions involving asynchronous operations (`onExecution`, `onFavScript`, `onDeleteScript`, `onEditScript`) to gracefully manage potential failures.

Second Version Review

1. **Error Handling in `handleStatusChange`:**
   - The `handleStatusChange` function lacks error handling to manage potential failures when updating the status.
2. **Optimizing Conditional Rendering:**
   - Store the value of `user?.isAdmin` in a constant to enhance readability and avoid multiple checks.
3. **TypeScript Types:**
   - Ensure that TypeScript types are explicitly defined and consistently used for better type safety and clarity.
4. **State Initialization for Script Statuses:**

- The state for managing script statuses (`scriptStatuses`) is introduced but not utilized efficiently. It might be better to rely directly on the `scripts` prop for the current status.

5. **Redundant Check for Empty State:**
   - The check for an empty script list is good, but it could be optimized to avoid rendering the entire table structure when it's not necessary.
6. **Simplifying State Management:**
   - The state management for script statuses could be simplified or even removed if the status is directly tied to the `scripts` prop, reducing complexity.
7. **Consistency in Prop Naming:**
   - Ensure consistency in naming props and variables to avoid confusion. For instance, `favScriptList` is named differently in different parts of the code.

Second Version - Improvement Methods:

1. **Incorporate Error Handling for Status Changes:**
   - Add error handling within the `handleStatusChange` function to manage potential failures when updating the status, ensuring a more robust application.
2. **Optimize Role-Based Rendering:**
   - Store the value of `user?.isAdmin` in a constant at the beginning of the component to avoid multiple checks and improve readability.
3. **Ensure Consistent TypeScript Usage:**
   - Define and use explicit TypeScript types for all script-related objects and props consistently to enhance type safety and code readability.
4. **Improve State Initialization for Status:**
   - Optimize state management for script statuses by either directly using the `scripts` prop or ensuring the state is initialized and used efficiently.
5. **Refine Empty State Logic:**
   - Optimize the logic for checking an empty script list to avoid rendering unnecessary table structures when the list is empty.
6. **Simplify Script Status State Management:**
   - Consider reducing complexity in state management for script statuses if the status can be directly derived from the `scripts` prop.
7. **Maintain Naming Consistency:**
   - Ensure consistent naming conventions for props and variables throughout the component to avoid confusion and maintain code clarity.

Code Review for Overview Page

Project Name/Module Name: Overview

Author: @Junhao KONG

Reviewer: @yijun liu

Date: May 14, 2024 May 26, 2024

Tools Used: Manual inspection, VS Code for syntax highlighting.

Code Review for the First Version:

Issue 1: Props Type Safety

Issue: The props `count`, `page`, and `rowsPerPage` are optional, which can lead to runtime errors if they are not provided.

Suggestion: Make these props required to ensure the component always receives the necessary data.

Issue 2: Code Cleanliness

Issue: There are multiple empty lines that reduce code readability.

Suggestion: Remove unnecessary empty lines to maintain a clean code structure.

Issue 3: Function Definitions

Issue: The `onExecution`, `onEditScript`, and `onDeleteScript` functions are not passed as props, reducing the component's flexibility.

Suggestion: Ensure these functions are passed as props from the parent component for better modularity and reuse.

Issue 4: Use of `user!.isAdmin`

Issue: Using a non-null assertion (`user!`) can lead to runtime errors if `user` is null or undefined.

Suggestion: Add a null check before accessing `isAdmin` to prevent potential crashes.

Next Actions:

- **Refactor Prop Types**: Make `count`, `page`, and `rowsPerPage` required props.
- **Clean Up Code**: Remove unnecessary empty lines.
- **Enhance Function Flexibility**: Ensure `onExecution`, `onEditScript`, and `onDeleteScript` are passed as props.
- **Improve Safety**: Add a null check for `user`.

Code Review for the Second Version：

Issue 1: Type Definitions

Issue: `ScriptTableData` is defined within the file, reducing reusability across the project.

Suggestion: Move `ScriptTableData` to a shared type definition file for reuse in other components.

Issue 2: Error Handling

Issue: Errors are handled using `alert` and `console.error`, which are not user-friendly.

Suggestion: Use `toast.error` to provide better user feedback in case of errors.

Next Actions:

- **Refactor Type Definitions**: Move `ScriptTableData` to a shared type definition file.
- **Improve Error Handling**: Use `toast.error` instead of `alert` and `console.error`.
- **Optimize Handler Functions**: Use `useCallback` and define handlers outside the component.

Code Review for the Active/ Inactive, Script-related APIs

Project Name/Module Name: Asynchronous Tasks Handling

Author: Lingyi Kong

Reviewer: Lingyi Kong

Review Date: May 15, 2024 May 22, 2024

Code Review for the first version

Authorization and Authentication:

- Validation: Ensure the authorization logic is robust and correctly verifies user permissions before allowing access or modifications. This includes checking if a user is an admin or has the necessary roles.
- Edge Cases: Consider scenarios where different user roles may have varying access rights, and handle these cases appropriately.

Error Handling:

- Specific Exceptions: Use specific exceptions instead of broad exception handling to make debugging easier and improve code robustness.
- Logging: Log all exceptions with sufficient detail to facilitate troubleshooting.
- Client Communication: Provide meaningful error messages and appropriate HTTP status codes to the client to ensure clarity and usability.

Code Reusability:

- Helper Functions: Avoid code duplication by creating helper functions for repetitive tasks, which will make the code easier to maintain and extend. For example, refactor the common operations in `admin_active_script` and `admin_inactive_script` into a single function.

Performance Considerations:

- Optimized Queries: Optimize database queries by using indexes and limiting the amount of data fetched. This can significantly improve performance.
- Pagination: Implement pagination for endpoints that return large datasets to enhance performance and usability.

Consistency:

- Coding Style: Maintain consistency in coding style, such as the use of single or double quotes for strings, and ensure consistent formatting and indentation throughout the codebase.

Documentation:

- Function Docstrings: Document each function with docstrings that describe its purpose, parameters, return values, and any exceptions raised.
- Module Documentation: Provide high-level documentation for the module, explaining its overall functionality and usage. This helps new developers understand the context and purpose of the code.

Detailed Feedback

`handle_list_scripts`

- Ensure proper handling of different user roles, providing comprehensive error messages and logging for debugging.
- Consider refactoring code to handle repetitive logic for fetching and formatting script data.

`active_script` and `inactive_script`

- Simplify the logic by creating a helper function for activating and deactivating scripts.
- Ensure all error scenarios are covered and logged properly.

`list_all_active`

- Ensure this function returns all active scripts efficiently and consider implementing pagination if the dataset is large.

`get_favorite_script`

- Ensure robust error handling and logging to capture any issues during the retrieval of favorite scripts.
- Optimize database queries and consider potential performance improvements.

`get_favorite_scripts`

- Ensure the function handles scenarios where the user or favorite scripts might not be found, providing clear and informative error messages.
- Log detailed information about the process to assist with debugging.

`admin_active_script` and `admin_inactive_script`

- Refactor common operations into a single helper function to reduce code duplication.
- Ensure comprehensive error handling and logging for all possible failure scenarios.

`list_active_scripts`

- Optimize the query to fetch active scripts and ensure the returned data is formatted consistently.
- Implement additional error handling and logging to capture any issues that might arise during execution.

Code Review for the second version

Authorization and Authentication:

- Ensure that the authorization logic is robust and correctly implemented. For instance, verify if users have the right permissions before allowing them to access or modify resources.
- Consider edge cases where different roles (e.g., admin, user) may have different access rights.

Error Handling:

- Use specific exceptions where possible instead of catching all exceptions. This will make debugging easier and the code more robust.
- Log all exceptions with sufficient detail to aid in troubleshooting.
- Provide meaningful error messages and HTTP status codes to the client.

Code Reusability:

- Avoid code duplication by creating helper functions for repetitive tasks. This will make the code easier to maintain and extend.
- Example: Refactor common operations in `admin_active_script` and `admin_inactive_script` into a single helper function.

Performance Considerations:

- Optimize database queries by using indexes and limiting the amount of data fetched whenever possible.

- Consider pagination for endpoints that may return large datasets.

Consistency:

- Maintain consistency in coding style, such as the use of single or double quotes for strings.
- Ensure consistent formatting and indentation throughout the codebase.

Documentation:

- Document each function with docstrings that describe its purpose, parameters, return values, and any exceptions raised.
- Provide high-level documentation for the module explaining its overall functionality and usage.

Code Review for upload and complete api

Module Name: Favorite function

Author: @Yuncong Ji

Reviewer: @Lingyi Kong

Review Date: 20/5

Code Review 1

Strengths:

1. **Modular Approach**: The code is well-structured and modular, separating the chunk saving and upload completion logic into distinct functions (`save_chunk` and `complete_upload`). This makes the code more maintainable and easier to test.
2. **Directory Creation**: The code checks for the existence of directories (`UPLOAD_TEMP_PATH` and `UPLOAD_PATH`) and creates them if they do not exist. This ensures that the file operations will not fail due to missing directories.
3. **Chunk Validation**: The `complete_upload` function checks for the existence of each chunk file before attempting to assemble the final file. This is a good practice to ensure data integrity.

Areas for Improvement:

1. **Error Handling**: The code currently does not handle exceptions that might occur during file operations (e.g., reading/writing files). Adding try-except blocks can help catch these errors and return more informative messages to the user.
2. **Security**: The filenames are directly used for creating file paths, which can lead to security vulnerabilities (e.g., path traversal attacks). It is advisable to validate and sanitize filenames before using them.
3. **Concurrency**: If multiple users upload files simultaneously, there could be conflicts with the temporary directory (`UPLOAD_TEMP_PATH`). Consider using unique temporary directories for each upload session to avoid such issues.

Code Review 2

Strengths:

1. **Clear API Endpoints**: The API endpoints are clearly defined, with routes for uploading chunks and completing the upload. This provides a straightforward interface for clients to interact with.
2. **Chunk-Based Upload**: Implementing a chunk-based upload mechanism is a robust way to handle large files, as it allows the upload process to resume from where it left off in case of a failure.
3. **Status Messages**: The functions return clear and informative status messages, which can be helpful for debugging and user feedback.

Areas for Improvement:

1. **Index Validation**: The `index` parameter is assumed to be valid and unique for each chunk. Adding validation to check for duplicate or out-of-order chunks can help ensure the integrity of the final assembled file.
2. **Logging**: Adding logging statements can help monitor the upload process and debug issues. For example, log the start and end of each chunk upload, any errors encountered, and the completion of the entire upload.

Handover

Handover section contains detailed instruction for client about hoe to deploy and set up the FL-Koala backedn and frontend environment, and final production demo and handover to client email log.

Final Product

Owned by Chloe_Duan • Updated 14 minutes ago

👋 Final Product Video Demo In case the following link doesn't work, please click Link to our video https://youtu.be/SKR6WB9w_GM?si=ph-DlBFy_Hgss_vK In case youtube link failed, we provide original demo video here. However, due to the limitation of size of uploading file on Confluence, please contact the following email for original video. ldduan@st

Confluence

Open preview

Deployment Guidance

Owned by Chloe_Duan • Updated 10 minutes ago

This document provides instructions on how to deploy and set up the FL-Koala backend and frontend environment. FL-Koala Backend Setup Guide This section includes instructions on how to set up the FL-Koala backend environment via Docker Compose. Prerequisites Python 3.12 Docker and Docker Compose (for Docker setup) PostgreSQL >=15.6 (for database se

Confluence

Open preview

Deployment Guidance

This document provides instructions on how to deploy and set up the FL-Koala backend and frontend environment.

FL-Koala Backend Setup Guide

This section includes instructions on how to set up the FL-Koala backend environment via Docker Compose.

> ⚠ **Prerequisites**
>
> - Python 3.12
> - Docker and Docker Compose (for Docker setup)
> - PostgreSQL >=15.6 (for database setup)

Setup Using Docker Compose

1. Build and Run with Docker Compose

Ensure you have Docker and Docker Compose installed. Open a terminal: Set the `DATABASE_URL` on your host machine:

- On Linux/Mac:

```
export DATABASE_URL="postgresql://yourusername:yourpassword@localhost:5432/yourdbname"
```

- On Windows (Command Prompt):

```
set DATABASE_URL="postgresql://yourusername:yourpassword@localhost:5432/yourdbname"
```

2. Ensure PostgreSQL is Running

Make sure PostgreSQL is installed and running on your server. If not installed, you can follow the installation guide for your specific operating system.

3. Inspect SQL Files

- `FL-Koala/src/backend/migrate/schema.sql`: Contains the SQL commands to create tables and schema.
- `FL-Koala/src/backend/migrate/seed.sql`: Contains the SQL commands to create sample scripts and users.

4. Run SQL Scripts

Before running the Docker containers, you need to initialize the database. You can do this by running the following commands:

- On Linux/Mac:

```
psql -U yourusername -d yourdbname -a -f schema.sql
psql -U yourusername -d yourdbname -a -f seed.sql
```

- On Windows (Command Prompt):

```
psql -U yourusername -d yourdbname -a -f schema.sql
psql -U yourusername -d yourdbname -a -f seed.sql
```

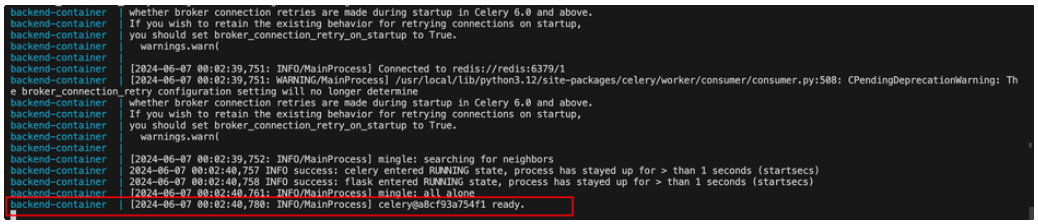
5. Run Docker Compose

Navigate to the project directory (`~/FL-Koala/src/backend`) and run:

```
docker-compose up --build
```

This command builds the Docker image and starts the service, exposing it on port 8080.

Docker is beginning to build, please wait

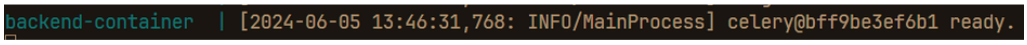


You will see the message when it runs successfully

Rebuilding the Image

If you need to rebuild the image, run:

```
docker-compose up --build
```



Suceessfully rebuilt

Summary

- Ensure PostgreSQL is running and accessible.
- Use `schema.sql` for schema setup and `seed.sql` for initial data.
- Properly set `DATABASE_URL`.

FL-Koala Frontend Setup Guide

Before setting up the frontend, ensure that your environment meets the following prerequisites:

> **Prerequisites**
>
> - Node.js >= 16.x
> - npm >= 8.x

You can download and install Node.js and npm here Node.js — Run JavaScript Everywhere

Package Managers

- It is recommended to use `yarn` or `pnpm` as the package manager
- Install yarn:

```
npm install --global yarn
```

- Install pnpm:

```
npm install --global pnpm
```

Setup Frontend

Navigate to (`~/FL-Koala/src/frontend`) and run commands:

1. Dependency Installation

Run the following command in the project root directory to install the required dependencies:

```
yarn install
```

Or using pnpm:

```
pnpm install
```

2. Run the server

```
yarn run dev
```

Or using pnpm:

```
pnpm run dev
```

Ready message poping-up means you successfully build frontend.

3. Open the browser

After “Ready” message pops up, navigate to `http://localhost:3000` in any local brower

Q&A:

Q: How can I install new python libraries that are used in scripts?

A: Update `~/FL-Koala\src\backend\requirements.txt` so that the backend service can run scripts properly.

Q: How can I clear up the input and output space?

A: Inspect `~/FL-Koala\src\backend\data` to remove unnecessary files.

Q: Why the building commond `docker-compose up --build` doesn’t work for me?

A: Please check if you navigate the right project directory `~/FL-Koala/src/backend`

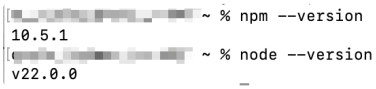
Common issues

Node.js Version Issues

If you encounter version mismatch issues, you can use `nvm` to manage and switch between different Node.js versions:

```
nvm install 10
nvm use 10
```

Current developer's environment version:



Final Product

👋 Final Product Video Demo

⚠️ In case the following link doesn't work, please click Link to our video

https://youtu.be/SKR6WB9w_GM?si=ph-DlBFy_Hgss_vK

> In case youtube link failed, we provide original demo video here. However, due to the limitation of size of uploading file on Confluence, please contact the following email for original video. ldduan@student.unimelb.edu.au

🤖 Final Presentation Slides

This is the final presentation slides containing introduction of Project Background, Product Showcase, Current Progress, Project Management, Project Reflection, Handover.

FL-Koala Final Presentation.pptx